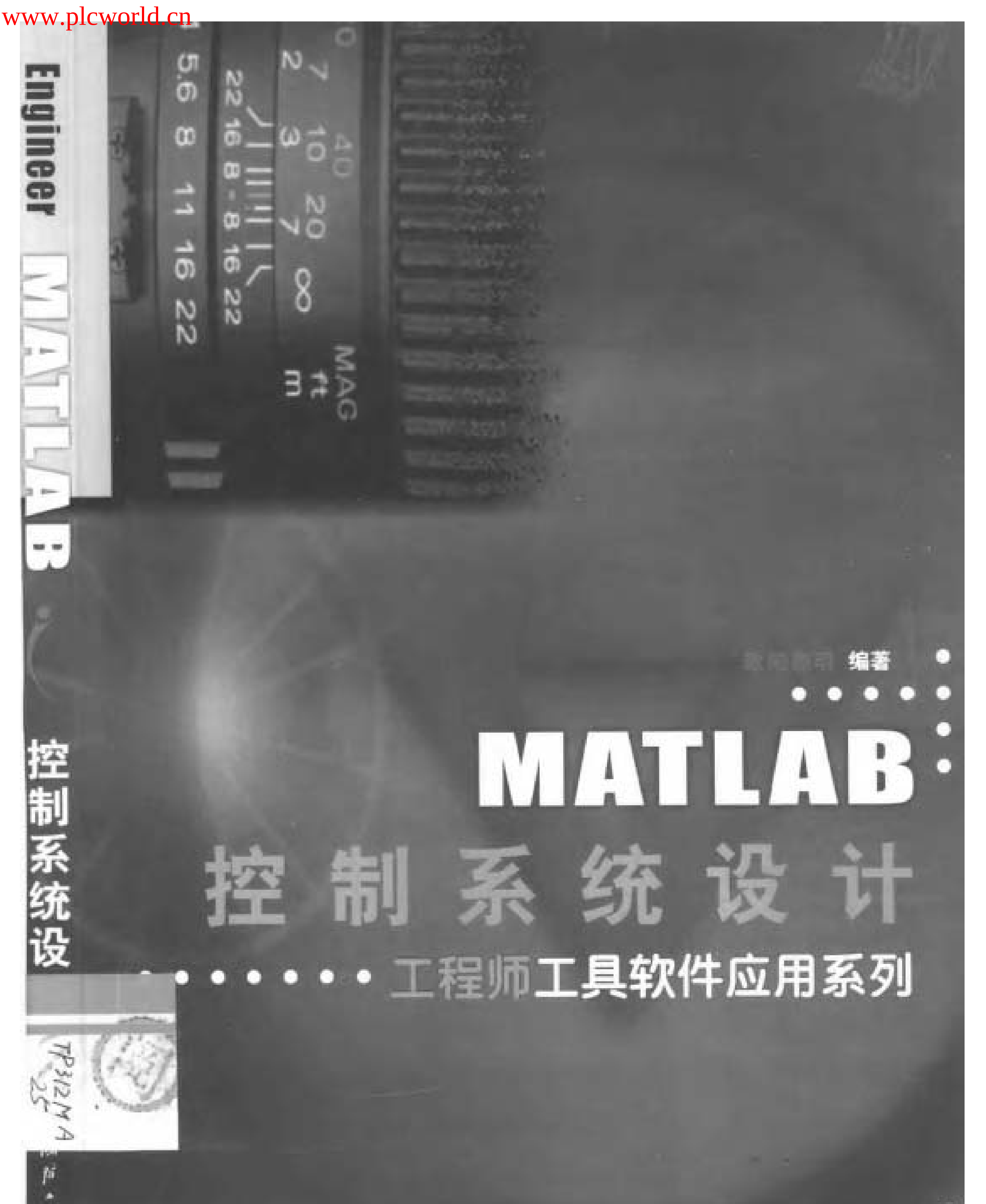
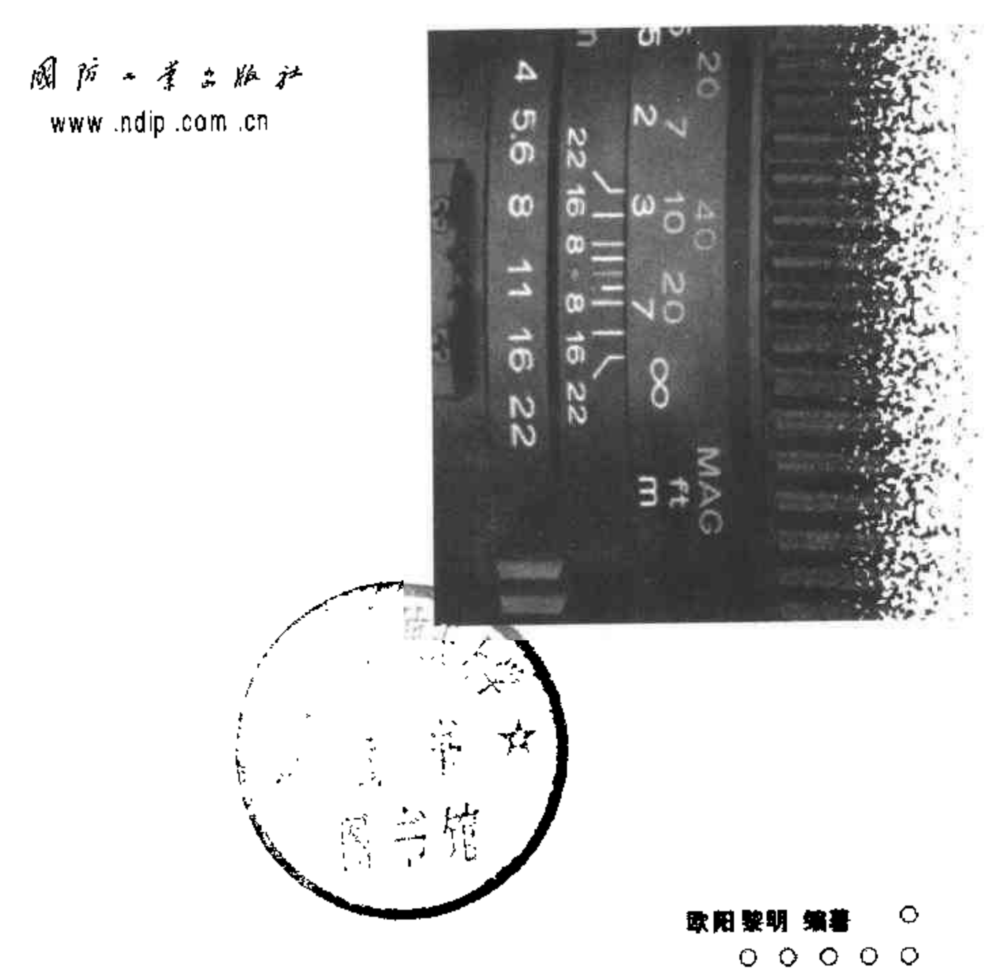

Engineer MATLAB
控制系统设
编著
MATLAB
控制系统设计
工程师工具软件应用系列
国防工业出版社



国防工业出版社
www.ndip.com.cn



欧阳黎明 编著

MATLAB 控制系统设计

工程师工具软件应用系列



内 容 提 要

全书以使用为经，以控制理论为纬，系统全面地介绍了国际控制界最为流行的计算机辅助设计及教学工具软件 MATLAB 在控制系统设计中的各项应用，包含了控制系统的数学描述、控制系统时频分析及根轨迹的绘制、传递函数模型控制系统校正、控制系统的状态空间设计方法、SIMULINK 技术以及神经网络与控制等内容。在附录中列举了一些常用的 MATLAB 命令和工具箱，方便读者查询，还补充了一些 MATLAB 网址，读者可以从这些地方随时获取关于 MATLAB 的最新信息。

本书旨在提高读者对控制系统的分析与设计能力，理论联系实际，适合于工科院校以及从事控制系统设计与仿真研究的师生和科研人员阅读使用，也可作为控制系统设计课程的教材。

图书在版编目(CIP)数据

MATLAB 控制系统设计/欧阳黎明编著 .—北京：国防工业出版社，2001.1

(工程师工具软件应用系列)

ISBN 7-118-02405-8

Ⅰ. M…　Ⅱ. 欧…　Ⅲ. 计算机辅助计算-软件包，MATLAB　Ⅳ. TP391.75

中国版本图书馆 CIP 数据核字(2000)第 49334 号

国防工业出版社出版发行

(北京市海淀区紫竹院南路 23 号)

(邮政编码　100044)

北京奥隆印刷厂印刷

新华书店经售

*

开本 787×1092　1/16　印张 21　486 千字

2001 年 1 月第 1 版　2001 年 1 月北京第 1 次印刷

印数：1—3000 册　定价：28.00 元

前　　言

第一版的 MATLAB 是 1985 年推出的 MATLAB1.0，其最初的目标是设计一种高效的解释性数学计算语言。因此，它将计算、可视化、编程等基本功能都集成在一个易于使用的环境中，并且问题的表达和求解都使用与日常数学运算相似的形式。MATLAB 的这种特点大大简化了工程技术人员熟悉开发环境和编制程序的时间，一些专业人员亲切地将 MATLAB 称作“演算纸式的语言”。当前，MATHWORKS 公司已经推出了最新版本的 MATLAB5.3，其运算功能和求解能力早已远远超出了最初的设计构想，但其友好的界面风格、人性化的求解过程依旧没有改变。MATLAB 的典型应用包括以下几个方面：

- 数学推导与数值计算
- 算法开发
- 建模与仿真
- 数据分析与可视化求解
- 科学与工程绘图
- 应用开发以及图形用户界面(GUI)功能

对于国际控制学术界来说，1985 年 MATLAB1.0 的推出更是一件划时代的大事。虽然起初该软件并不是为控制系统设计的，但它提供了强大的矩阵处理和二、三维图形绘制功能，具有较高的可信度和灵活的使用方法，非常适合现代控制理论的计算机辅助设计。事实上，具有较高的可信度和灵活的使用方法，非常适合现代控制理论的计算机辅助设计。事实上，MATLAB 进行科学和工程计算的核心元素是矩阵，并且不限定维数，而现代控制理论正是建立在矩阵运算的基础之上的。控制理论领域的研究人员正是注意到这一点，在 MATLAB 提供的基础上开发了许多与控制理论相关的程序集，这些程序集目前都作为工具箱(Toolbox)集成在 MATLAB 环境里。例如控制系统工具箱(Control System Toolbox)、鲁棒控制工具箱(Robust Control Toolbox)、系统辨识工具箱(System Identification Toolbox)等等，这些工具箱都是理论水平和实际经验非常深厚的控制界权威主持开发的，与控制理论的学习和应用都结合的非常紧密，无论是控制专业的学生还是经验丰富的控制工程师，都能从中找到有用的东西。

1993 年 SIMULINK1.0 问世，这个集成在 MATLAB4.0 中的动态系统建模、仿真工具使控制系统的计算机辅助设计向可视化的方向迈进了一大步。从此，控制系统建模与仿真摆脱了烦琐的关联矩阵求取和输入，可以将更多的精力集中在系统的设计和校正上，这尤其是控制理论专业学生的福音。事实上，国际自控联(IFAC)第四次控制教育研讨会上就以把学生的软件技能和实际动手能力列为教育目标之一。而世界各国在控制理论的教学中，均采用 MATLAB 及

SIMULINK 作为辅助教学软件，一方面可以摆脱繁杂的大规模计算，另一方面还可以使学生有机会自己动手构建模型，所花的代价要远小于实际建模。

目前 SIMULINK 最新的版本是 3.0，集成在 MATLAB 5.3 环境中，该版本一方面集成了原有 SIMULINK 的可视化的优点，另一方面还添加了诸如模块优先权与整合、总线模块、数据可视化处理等新的功能，为 SIMULINK 在控制领域的应用开辟了更加广阔的前景。

MATLAB 还提供了一系列的控制语句，这些语句的语法和使用规则都类似 FORTRAN、C 等高级语言，但比高级语言更加简洁。并且，由于 MATLAB 拥有种类繁多、功能丰富的函数及工具箱，因而在程序编制过程中，几乎不必考虑具体的数值计算的算法实现问题。在数值计算方面，MATLAB 的另一个优点就是拥有良好的数值稳定性，对于病态或非正则矩阵一般都能给出正确的解。MATLAB5.3 中还集成了 C/C++编译器，也就是说，在 MATLAB 环境下可以调用已经调试通过的 C/C++程序或库函数，这在一定程度上拓展了 MATLAB 的应用领域。

最后还应该指出，MATLAB 是与机器类型无关的。也就是说，在一台计算机上用纯粹的 MATLAB 语言编制的程序可以不做任何修改直接拷贝到另一台计算机上运行，当然，这台计算机上要拥有 MATLAB 环境。目前，MATLAB 可以在下列各类计算机上运行：IBMPC 及其兼容机、Macintosh、Sun 工作站、VAX/Alpha 机、Apollo 工作站、HP 工作站、DEC station 工作站、SGI 工作站、RS/6000 工作站、Convex 工作站及 Gray 计算机等，这些机器基本上包括了当前的常用机型。MATLAB 的这些特性都是为了让使用者不必过多纠缠于计算机的硬件的低级细节和具体的程序实现问题，而将更多的精力放在实际问题的解决上去。

本书的主要内容是 MATLAB 在控制系统与仿真领域的具体应用，其重点是基本概念的阐述和基本方法的讲解，并不过多地纠缠理论问题的推导。因此，各章都以实例为主线，包括背景介绍、问题描述、分析求解、题后小结等各方面内容。希望能够对本领域感兴趣的读者有所裨益。同时，读者可在配套光盘中调用全部程序。

全书由欧阳黎明编著，张强、李东、丁岩、刘春晖、程国云、杜志华、刘松、王炎、谭思其、张伟、宋昆、王华等同志参与了本书的编写工作，张辉、李洪、杜海涛、刘为、刘英等同志调试并能与了部分程序的编制工作，在此向所有为本书的出版提供了无私帮助的同志致以衷心的谢意。

由于时间仓促，限于作者的水平，本书的不足之处在所难免，欢迎广大读者不吝赐教。

编　者

2000 年 9 月

目　录

第一章　MATLAB 初探

1.1　MATLAB 简介

1985 年 MATLAB 1.0 推出后，立刻受到了国际控制学术界的重视。虽然起初该软件并不是为控制系统设计的，但它提供了强大的矩阵处理和绘图功能，可信度高，灵活方便，非常适合现代控制理论的计算机辅助设计。控制理论领域的研究人员正是注意到这一点，在其基础上开发了许多与控制理论相关的程序集，这些程序集目前都作为工具箱(Toolbox)集成在 MATLAB 环境里。例如：控制系统工具箱(Control System Toolbox)、鲁棒控制工具箱(Robust Control Toolbox)、系统辨识工具箱(System Identification Toolbox)等等，与控制理论的学习和应用都结合的非常紧密，无论是控制专业的学生还是经验丰富的控制工程师，都能从中找到有用的东西。

1993 年 SIMULINK 1.0 问世，这个集成在 MATLAB 4.0 中的动态系统建模、仿真工具使控制系统的计算机辅助设计向可视化的方向迈进了一大步。从此，控制系统建模与仿真摆脱了烦琐的关联矩阵求取和输入，可以将更多的精力集中在系统的设计和校正上，这尤其是控制理论专业学生的福音。事实上，国际自动控制联合会(IFAC)第四次控制教育研讨会上就以把学生的软件技能和实际动手能力列为教育目标之一。而世界各国在控制理论的教学中，均采用 MATLAB 及 SIMULINK 作为辅助教学软件，一方面可以摆脱繁杂的大规模计算，另一方面还可以使学生有机会自己动手构建模型，所花的代价要远小于实际建模。

MATLAB 还提供了一系列的控制语句，这些语句的语法和使用规则都类似 FORTRAN、C 等高级语言，但比高级语言更加简洁。并且，由于 MATLAB 拥有种类繁多、功能丰富的函数及工具箱，因而在程序编制过程中，几乎不必考虑具体的数值计算的算法实现问题。在数值计算方面，MATLAB 的另一个优点就是拥有良好的数值稳定性，对于病态或非正则矩阵一般都能给出正确的解。最后还应该指出，MATLAB 是与机器类型无关的。也就是说，在一台计算机上用纯粹的 MATLAB 语言编制的程序，可以不做任何修改直接拷贝到另一台计算机上运行，当然，这台计算机上要拥有 MATLAB 环境。目前，MATLAB 可以在下列各类计算机上运行：IBM PC 及其兼容机、Macintosh、Sun 工作站、VAX/Alpha 机、Apollo 工作站、HP 工作站、DEC station 工作站、SGI 工作站、RS/6000 工作站、Convex 工作站及 Gray 计算机等，这些机器基本上包括了当前的常用机型。MATLAB 的这些特性都是为了让使用者不必过多纠缠于计算机的低级细节和具体的程序实现问题，而将更多的精力放在实际问题的解决上去。

当前，MATLAB 已经成为国际控制界最为流行的计算机辅助设计及教学工具软件，虽然它可能已偏离了“矩阵实验室”(Matrix Laboratory)的本意，但其风格和设计理念对

控制领域的影响是非常深远的。本书以 MATLAB 的使用为经，控制理论为纬，希望读者通过阅读本书能够对 MATLAB 在控制系统中的应用有一定了解，提高控制系统的分析与设计能力。

本章的内容主要是最基础的 MATLAB 集成环境及使用，一方面是希望能给没有接触过 MATLAB 的控制领域专业人员以入门性指导，另一方面也是为了保持内容的完整性。熟悉 MATLAB 的读者可以跳过本章，直接从第二章开始，需要的时候再返回来查一查相关的概念。

1.1.1　MATLAB 环境

目前最新的版本是 MATLAB 5.3，但大部分研究人员都还在使用 MATLAB 5.1 或 5.2，在教学实验室中还有大量的 MATLAB 4.x 存在。事实上，从 1985 年 MATLAB 1.0 出现发展至今，其语法结构和使用方法都基本类似，新增的只是各种功能函数和繁多的工具箱。因此，本书的叙述和讲解基本上以 MATLAB 5.x 为基准，兼顾 MATLAB 4.x。各版本之间的具体差别，请参阅 MATLAB 帮助及其 Release Note。本书附录中罗列了一些与 MATLAB 相关的网址，读者还可以通过国际互联网随时查阅有关版本的最新消息。

在 Windows 95/98 系统环境下安装好 MATLAB 5.x 后，双击 MATLAB 图标或从“开始”菜单打开 MATLAB，即可进入 MATLAB 集成环境，首先是 MATLAB 命令窗(MATLAB Command Window)，如图 1-1 所示。

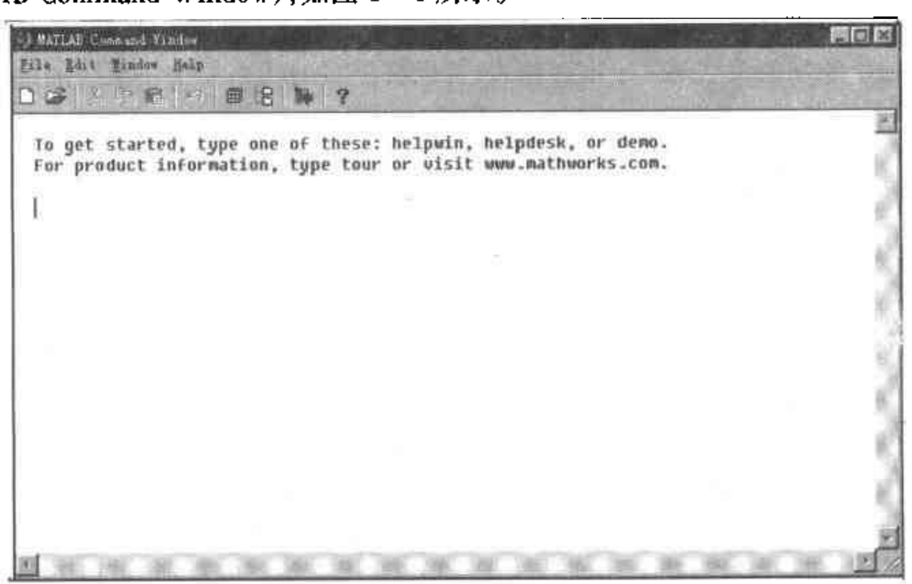


图 1-1　MATLAB 命令窗

在 MATLAB 4.x 版本中该界面下有提示符“> >”，5.x 的版本省略了这一提示符。具体的命令、函数调用等都在此窗口下进行。为了节省版面，以后的例子不再出现此窗口，仅以 MATLAB Command Window 代替。

MATLAB for Windows 的各版本集成环境的风格都大体类似，熟悉 Windows 的读者可以很快掌握其使用方法，在此不再赘述。

在 MATLAB Command Window 环境下键入“tour”可以进入 MATLAB Tour 浏览程

序,这是一个很好的图形化 MATLAB 交互学习软件,包括 MATLAB、SIMULINK、Stataflow、Toolbox、Programming、Dynamic System Simulation、Control Design、Signal Processing 等各方面内容。在 MATLAB Command Window 环境下键入"demo"可以进入 MATLAB Demos 演示程序,通过不同的例子,读者可以深入学习 MATLAB 的各项功能。

与 MATLAB 4.x 相比,MATLAB 5.x 有了自己的 M 文件编辑器——MATLAB Editor/Debugger,不必再使用 Note Pad 或别的文字处理软件。在 MATLAB Command Window 环境下选择"File"菜单 – "New" – "M – file"或直接键入"edit"即可进入 MATLAB Editor/Debugger,如图 1 – 2 所示。

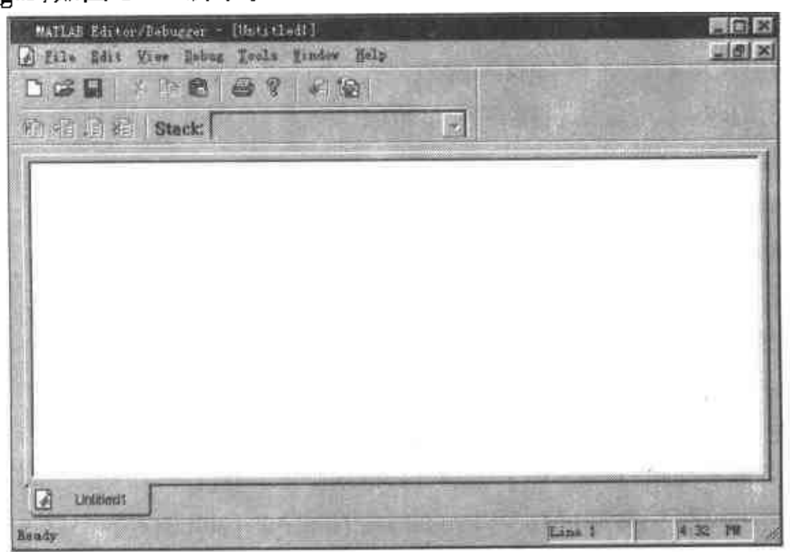


图 1 – 2 MATLAB Editor/Debugger 编辑器

这个编辑器与高级语言的集成开发环境非常类似。除了常用的文件管理系统和文字处理功能外,编写好的 M 文件在此环境下可以进行修改、调试、跟踪、设置清除断点、单步执行或按指定的条件执行。执行过程中的错误和警告都直接写在 MATLAB Command Window 中,保持此编辑器窗口的整洁。

1.1.2 MATLAB 主要功能

MATLAB 发展至今,已不仅仅是单纯矩阵运算的数学处理软件,其开放式的结构吸引了许多优秀人才编写 M 函数和工具箱,目前已经渗透到了工程计算和设计的各个领域。其中与控制系统设计与仿真相关的功能大致有以下各项:

1. 数值计算及分析

(1) 向量、矩阵的运算分析。

(2) 复数运算及分析

(3) 微分方程的求解

(4) 稀疏矩阵的运算

(5) 特殊函数的计算机分析

(6) 快速傅里叶变换及信号处理矩阵计算

(7) 数据分析及统计计算

2. 程序语言及算法实现
(1) 程序流程控制语句
(2) C 语言产生器
(3) 文件管理及二进制输入输出
3. MATLAB 绘图功能
(1) 二维图形绘制
(2) 特殊坐标图形绘制及修改
(3) 三维坐标图形绘制
4. MATLAB 与高级语言接口
(1) MATLAB 与 C 语言接口及库函数
(2) MATLAB 与 C++语言接口及库函数
(3) MATLAB 编译器
5. Simulink 建模与仿真
(1) Simulink 加速器
(2) 实时工作空间(Real-Time Workshop)
(3) 非线性控制系统设计
6. 相关工具箱
(1) 控制系统工具箱(Control System Toolbox)
(2) 鲁棒控制工具箱(Robust Control Toolbox)
(3) 模型预测控制工具箱(Model Predictive Control Toolbox)
(4) mu 分析与校正工具箱(mu Analysis and Systhesis Toolbox)
(5) 多变量系统频域设计工具箱(Multi-Variable Frequency Design Toolbox)
(6) 定量反馈控制工具箱(Quantitative Feedback Theory Toolbox)
(7) 频域系统辨识工具箱(Frequency Domain System Identification Toolbox)
(8) 系统辨识工具箱(System Identification Toolbox)
(9) 神经网工具箱(Neural Network Toolbox)
(10) 小波分析工具箱(Wavelet Toolbox)
(11) 最优化工具箱(Optimization Toolbox)
(12) 偏微分方程工具箱(Partial Differential Equation Toolbox)
(13) 信号处理工具箱(Signal Processing Toolbox)
(14) 图像处理工具箱(Image Processing Toolbox)
(15) 扩展符号数学工具箱(Extended Symbolic Math Toolbox)

上述各项功能基本上构成了 MATLAB 在控制系统设计与仿真领域应用的主线，本书以下各章节的论述也基本上围绕这个主线进行。

1.2 基本语句结构

从根本意义上说，MATLAB 并不是一种真正的计算机语言，因为单纯的 M 函数无法产生可执行文件，脱离了 MATLAB 环境就无法执行。但从实际效果来看，MATLAB 语

言能够完成高级计算机语言的绝大部分功能,语法和语句结构也非常类似,因此,有高级语言基础的读者能够很快适应 MATLAB 的风格。

1.2.1 变量

作为 MATLAB 的最基本运算单元,变量是具体运算和编写函数的基础。与高级语言类似,变量的基本赋值语句结构是:

变量名 = 表达式

变量名可以是字母或数字,但首字符必须是字母。表达式可以是任意合法的数字、函数及运算符。如果表达式没有命名,则 MATLAB 把结果存储和显示在"ans"变量中,以备查询和使用。请看下例:

[例 1.1] 在 MATLAB 环境下求解表达式:

$$|e^{0.5}| + \sin(\frac{\pi}{6}) \times \ln(\sqrt{10})$$

分析:MATLAB 提供了种类繁多的基本数学函数,用法也与高级语言类似,直接调用即可。

求解过程:在 MATLAB Command Window 下直接键入:

```
x = exp(0.5) + sin(pi/6) * log(sqrt(10))
```

即可得到:

```
x =
      2.2244
```

若直接输入表达式,有:

```
exp(0.5) + sin(pi/6) * log(sqrt(10))
ans =
2.2244
```

通过上述例子可以很快熟悉 MATLAB 中变量的使用方法,事实上,通过这个例子还会发现 MATLAB 可以作为一个功能强大的函数计算器使用。关于变量的使用还有一些需要说明的问题:

(1) MATLAB 对变量名的大小写敏感,大小写代表不同的变量。

(2) 如果表达式最后一个符号是分号";",则表达式同样运行,但不在屏幕上显示结果。如果是变量名等于表达式,输入变量名可以查看结果。

(3) 显示结果的缺省格式是 5 为有效数字,命令"format short e"、"format long"和"format long e"的输出格式分别为 5 位浮点数、15 位定点数和 15 为浮点数。"fprintf"命令可以在屏幕上显示用户要求的格式,其具体格式与 C 语言类似,如:

```
fprintf('answer is %8.5f \ n',pi * 5.5^2)
answer is 95.03318
```

(4) MATLAB 可以响应键盘输入,"input"命令等待用户从键盘输入的数据:

```
x = input('input the variable x: ')
input the variable x: 1.1
x  = 1.1000
```

(5) MATLAB 中有一些保留的常量,例如 inf 表示无穷大。MATLAB 依照 IEEE 的

标准允许除数为 0,产生这种情况时只给出警告,不中止程序,结果为 inf

```
1/0
Warning: Divide by zero.
ans = Inf
```

需要注意的是如果给 inf 另外赋值,那么它将不再是无穷大,必须用 clear 命令清除所有变量后才恢复原来的意义。

另一个常量 NaN 表示非数字(Not a Number),一般是由 inf/inf 或 0/0 产生的。这些特点使得 MATLAB 比一般的高级语言有着更高的容错性,更加灵活可靠。

1.2.2 向量运算

向量是线性代数中最基本的概念,也是 MATLAB 绘图的基础元素。MATLAB 提供了多种指定和自动生成行或列向量的方法,符合人们的自然习惯。向量赋值的基本结构与变量的赋值相似:

```
x = [1  2  3]
x = 1    2    3
y = [4;5;6]
y = 4
    5
    6
```

如果向量中的元素用空格或逗号隔开,代表行向量,如果用分号隔开则代表列向量。向量自身可以求转置、范数,同维的向量可以进行加、减、乘和求内积的运算。乘法需遵守特殊规则,标量与向量相乘即标量与向量的每个分量相乘;向量间点乘为对应元素相乘。向量的范数采用欧几里德范数,定义为向量本身内积的平方根。为了避免烦琐枯燥的叙述,下面以一个实例简要说明上述概念。

向量 x = [1 2 3], y = [2 4 6],求向量 x 和 x 与 y 点乘所得向量之间的夹角。

分析:两向量 X、Y 之间的夹角定义为:

$$\cos\theta = \frac{\langle X, Y \rangle}{\| X \| \cdot \| Y \|}$$

可以通过上述 MATLAB 提供的向量运算指令及函数完成。

求解过程:在 MATLAB Command Window 下键入下述语句:

```
x = [1 2 3];
y = [4;5;6];
z = x.*y';
u = x*z';
alpha = acos(u/(norm(x)*norm(z)))
```

输出结果为两向量间夹角的弧度值:

```
alpha =
   0.1113
```

其余的各向量值为:

```
z = 4    10    18
u = 78
```

小结：从本例可以看出，MATLAB 中向量的赋值、运算和操作与在纸上直接书写算式没有什么区别，因此，MATLAB 也被称为“演算稿语言”。这一特点使得 MATLAB 语句非常易于理解，用户在解题过程中也可以集中精力考虑问题的实际算法，忽略数据结构的具体实现。

关于向量的赋值和运算，还有几点需要说明的地方：

1. 零向量的生成

控制理论中经常要用到向量元素均为零的所谓零向量，MATLAB 提供了一条零向量生成语句 zeros( )，其实现形式如下：

```
z = zeros(1,3)
z =      0     0     0
```

2. 各元素均相同的非零向量

MATLAB 提供了一条语句 ones()，可以生成各元素均为 1 的向量。对于各元素均相同的非零向量，只需将其乘以相应的系数即可：

```
p = 2 * ones(1,3)
p =  2   2   2
```

3. 元素为等差数列的向量

在控制系统仿真或进行迭代运算时，需要较长的元素为等差数列的向量来模拟时间轴或迭代数据，可以通过下述语句来实现这个功能：

```
x = 0:pi/3:2 * pi
x =   0    1.0472    2.0944    3.1416    4.1888    5.2360    6.2832
```

表示从 0 到 2π 之间每隔 $\pi/6$ 取一个点，组成一个 7 维向量。步长可以设置为任意实数，如果是负值就从大到小依次递减，如果省略步长设置，则表示默认步长为 1。如：

```
x = 1:7
x =      1     2     3     4     5     6     7
```

1.3 基本矩阵运算

以矩阵运算为基础的线性代数是整个现代控制理论的基石，也正是线性代数严密的数学基础，使得现代控制理论获得了飞速发展。事实上，MATLAB 最初就是作为“矩阵实验室”来开发的，相对于其他一些数学建模与仿真软件，它在这方面的优势是不言而喻的。考虑到读者的阅读习惯和循序渐进的原则，本章在内容安排上采用了标量——向量——矩阵这一符合线性代数思路的讲解顺序，但是在 MATLAB 语言环境中，最基本的元素是矩阵。标量是 1×1 的矩阵，向量是单行或单列的矩阵，虽然 MATLAB 还有其他存储数字或非数字数据的方法，但对于初学者来说，把所有数据均当作矩阵的做法是很有裨益的。与计算机高级语言不同的是，在 MATLAB 语言环境中可以分别对矩阵的整体和单个元素进行操作，这就使得使用 MATLAB 进行矩阵运算时更接近人们实际解决问题的思路，从而更加直观、方便。有了上一节变量和向量的基础，矩阵运算的指令和函数是很容易理解的。

1.3.1 矩阵的逆

与向量的生成类似，MATLAB环境下矩阵的生成也是由方括号中的一组数值或表达式实现的，行与行之间由分号";"隔开，行之间的元素用空格或逗号隔开。维数相同的矩阵可以进行加法、减法和乘法运算，其定义为矩阵的对应元素相加减。维数不同的矩阵进行加减运算时，MATLAB将给出错误信息。同样，维数不匹配的矩阵进行乘法运算时，MATLAB也会给出错误提示信息。单个矩阵可以进行转置和翻转运算，其定义与线性代数中完全一致，具体操作在后面的例子中有描述。

1. 直接求逆

矩阵求逆是控制系统中单个矩阵最重要的运算之一，无论是求解状态方程还是系统辨识，都涉及到高维数矩阵求逆的问题。在高级语言中，有许多较为成熟的算法，这些算法的数值稳定性问题也作过详细讨论，但都存在一定的局限性，需要使用者对算法有较深入的了解，才能保证获得满意的结果。而MATLAB中矩阵求逆只需一条简单的inv()语句即可完成，屏蔽了具体的算法细节，又保证了数值稳定性。MATLAB中还定义了矩阵的左除和右除。"\"运算符表示左除，A \ B就是由高斯消元法求解线性方程组 AX = B 的解，即A的逆左乘B；"/"运算符表示右除，B/A就是B左乘A的逆。下面给出一个例子，具体实践并检验上述各矩阵的基本运算。

对于下述线性方程组 AX = b，试用最小二乘法求取未知的参数向量X，并与其余几种方法相比较。

$$\begin{pmatrix} 4 & -2 & -10 & 2 \\ 2 & 10 & -12 & 5 \\ -4 & -6 & 16 & -6 \\ 3 & -2 & 8 & 7 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} -10 \\ 32 \\ 16 \\ 21 \end{pmatrix}$$

分析：最小二乘法是系统辨识领域的经典辨识方法，一般用于大规模数据的系统参数辨识。A是长方阵时(即行数远大于列数)，可求得：

$$X = (A^TA)^{-1}A^Tb$$

由于空间有限，本题的A矩阵并非长方阵，这里采用最小二乘法主要是为了使读者熟悉MATLAB基本矩阵运算。

求解过程：

(1) 最小二乘法

在MATLAB Command Window下键入下述语句：

```
A=[4 -2 -10 2;2 10 -12 5;-4 -6 16 -6;3 -2 8 7];
b=[-10;32;16;21];
X=inv(A'*A)*A'*b
```

键入回车后可得下述结果：

```
X = 42.5230
    17.7071
    10.1213
   -21.7322
```

(2) 左除求逆

在 MATLAB Command Window 下键入下述语句：

```
A=[4 -2 -10 2;2 10 -12 5;-4 -6 16 -6;3 -2 8 7];
b=[-10;32;16;21];
X=inv(A)*b
```

键入回车后可得下述结果：

```
X=  42.5230
    17.7071
    10.1213
   -21.7322
```

小结：在这里可以看到通过最小二乘法求解线性方程组和直接求逆得到的结果是一样的，这主要是由于 A 矩阵是方阵并且满秩的缘故。一般说来，系数矩阵是方阵的线性方程组可以通过左除或直接求逆来求解，如果系数矩阵满秩则可求得惟一解，否则可以求得维数不同的解空间。如果系数矩阵是长方阵，那么直接求逆就无能为力了，通过最小二乘法可以求得满意的结果。

事实上，系数矩阵不是方阵时，左除运算使用的算法就是最小二乘法。当然，这里使用的只是最基本的最小二乘法，仅考虑了矩阵 b 的扰动，没有考虑系数矩阵 A 的扰动，并不能保证所有情况下的数值稳定性。需要进一步深入的读者请参阅有关总体最小二乘法的书籍。

2. 矩阵的广义逆

对于非满秩的奇异矩阵或长方阵，也可以定义一种“逆”矩阵，这主要是为了求解线性方程组的方便。这里定义的“逆”矩阵，一般是最小范数意义上的，即满足：

$$\min_{X} \| AX-b \|$$

的向量 X。对于长方阵 A，这样的解有无穷多个。通常定义的 Moore-Penrose 广义逆矩阵满足下述条件：

(1) AMA = A

(2) MAM = M

(3) AM 和 MA 均为对称阵

在这些条件下可以证明，对于任意给定的矩阵 A，其广义逆 M 是惟一的。对于奇异的方阵 A，只需满足第一个条件 AMA = A，所求得的矩阵 M 也称作 A 的广义逆矩阵。一般说来，广义逆矩阵与原矩阵的乘积不再是单位矩阵，但它保留了逆矩阵的形式，即 AMA = A 和 MAM = M，并且在范数意义下是最小的，这对于解超定线性方程组尤其有用。请看下例：

将上例的矩阵 A 扩充为 8×4 的长方阵，相应的 b 矩阵扩充为 8 维向量，扩充的数据如下：

$$A=\begin{pmatrix} -2 & 12 & 4 & 8 \\ -1 & 11 & 6 & 11 \\ 2 & -4 & 12 & 7 \\ 11 & -8 & 2 & 15 \end{pmatrix} \quad b=\begin{pmatrix} -6 \\ -26 \\ -17 \\ 21 \end{pmatrix}$$

[例 1.2] 试分别用最小二乘法和求广义逆的方法求解此线性方程组,并比较其范数。

分析:对于上述超定线性方程组,直接求逆法是无法求出结果的,精确满足上述方程的向量 X 是不存在的,只能通过最小二乘法或求广义逆的方法求得其最小范数解。MATLAB 提供了求广义逆的函数 pinv(),可以直接使用。只要没有使用 clear 命令,本题的求解可以直接在上例的 MATLAB Command Window 中进行。

求解过程:

(1) 最小二乘法

在上例的 MATLAB Command Window 中直接键入下述语句:

```
A=[A;-2 12 4 8;-1 11 6 11;2 -4 12 7;11 -8 2 15];
b=[b;-6;-26;-17;21];
X=inv(A'*A)*A'*b;
Norm(A*X-b);
t=1:8;
plot(t,b,t,A*X,'+')
```

结果为:

```
X = 42.7264 17.7810 10.1603 -21.8219
norm(A*X-b)= 0.6771
```

结果仿真图如图 1-3 所示。

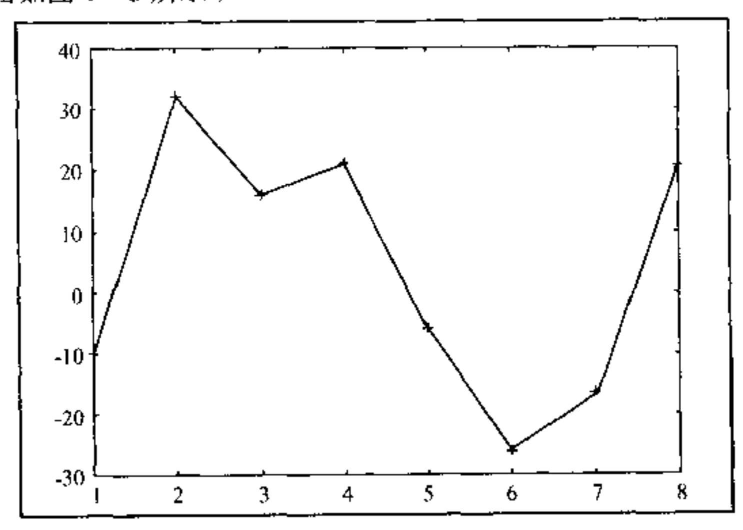


图 1-3 最小二乘法解线性方程组结果仿真

(2) 广义逆法

紧接着上一问求解,在同样的 MATLAB Command Window 中键入下述语句:

```
X=pinv(A)*b;
norm(A*X-b);
plot(t,b,t,A*X,'O')
```

结果为:

```
X = 42.7264 17.7810 10.1603 -21.8219
norm(A*X-b)= 0.6771
```

结果仿真图如图 1-4 所示。

小结:plot()语句是 MATLAB 中的二维图形绘制指令,不熟悉的读者不必深究其语法和结构,具体细节将在以后有关绘图的章节中详细论述。结果仿真图中,横轴是序号或时间轴,纵轴是向量元素的值。实线是向量 b 的值,"+"和"O"是由最小二乘法和广义逆法求得的解。从图中可以看出,解向量与系数矩阵之积与期望的 b 向量非常接近,在系统辨识或滤波器设计中这就说明事先估计的模型或滤波器系数是有效的。二者之差的范数值仅为 0.6771,这从另一个侧面说明解向量与精确值相差很小,最小二乘法和广义逆法在求解超定线性方程组时都是有效的。当然,绝对的精确解是求不到的,因为方程本身是超定的,只能求得某一约束条件下的最优解。

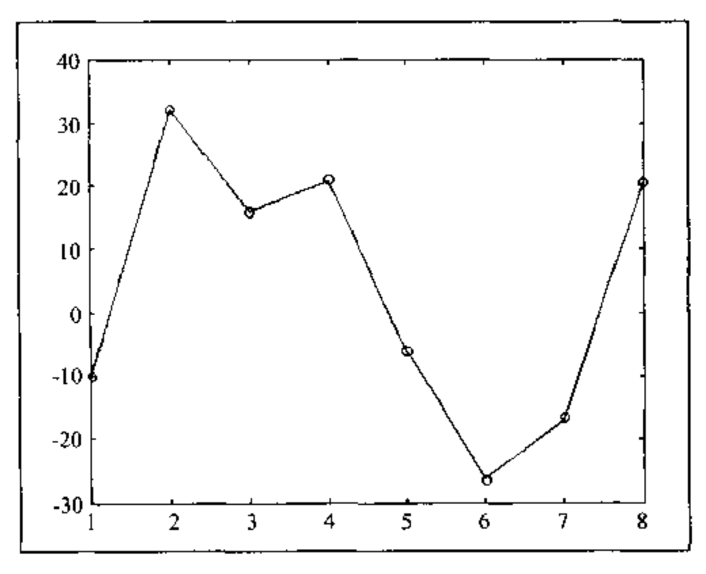


图 1-4　广义逆法求解线性方程组结果仿真

1.3.2　矩阵的三角分解

矩阵的三角分解就是将一个满秩方阵分解为一个下三角矩阵和一个上三角矩阵的乘积,又称为矩阵的 LU 分解。高级语言中一些求解线性方程组的算法就是基于矩阵的 LU 分解,因为,相对来说上三角和下三角矩阵的逆矩阵是比较容易求解的,并且如果注意到主对角元选取的问题,就可以保证计算的数值稳定性。在系统辨识的阶次推导和 Hankel 矩阵的理论推导过程中也会用到 LU 分解。

LU 分解具体的数值算法在任何一本线性代数的教科书中都可以查到,这里就不再赘述。MATLAB 提供了一条矩阵 LU 分解的函数 lu()可以很方便的求得变换后的下三角矩阵 L 和上三角矩阵 U。该函数考虑了主对角元选取的问题,有较好的数值稳定性,用户可不考虑具体的实现细节,放心的使用。需要注意的是,在某些情况下,该函数求得的 L 矩阵并不一定是下三角矩阵,因为在求解过程中可能进行了元素的行交换,即主对角元不再为 1。如果希望得到真正的下三角矩阵,就需要用到 MATLAB 存储行交换信息的交换阵 P(Permutation),此时不再有 X = LU,而是 PX = LU。我们通过下面这个例子熟悉 lu()函数的使用,并验证上述说法。

线性方程组 AX = b 如下,试用系数矩阵 LU 分解求解向量 X,并验证 lu()函数的各项功能。

$$\begin{pmatrix} 2 & 4 & 6 & 8 \\ 1 & 3 & 5 & 7 \\ 2 & 4 & 5 & 7 \\ 1 & 8 & 6 & 3 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} 10 \\ 11 \\ 12 \\ 13 \end{pmatrix}$$

分析：形如 AX = b 的线性方程组是控制系统设计、系统辨识、信号处理、滤波器设计等一系列控制理论相关领域中最基本的问题，其解法及解法的数值稳定性问题一直是控制领域的基本研究方向之一。事实上，本书一再讨论其解法问题就是希望给读者建立起这样的概念。当然，在使用 MATLAB 的过程中用 LU 分解来求解线性方程组的情况并不多见，这里是为了使读者熟悉 LU 分解函数的基本用法。

求解过程：

第一种格式：[L, U] = lu(A)

在 MATLAB Command Window 中键入下述语句：

```
A = [2 4 6 8;1 3 5 7;2 4 5 7;1 8 6 3];
b = [10;11;12;13];
[L,U] = lu(A);
X = inv(U) * inv(L) * b
```

结果为：

```
X = -4.4000 5.4000 -6.6000 4.6000
```

下三角矩阵 L：

```
L = 1.0000      0         0         0
    0.5000      0.1667    1.0000    0
    1.0000      0        -0.6667    1.0000
    0.5000      1.0000    0         0
```

上三角矩阵 U：

```
U = 2.0000    4.0000    6.0000    8.0000
    0         6.0000    3.0000   -1.0000
    0         0         1.5000    3.1667
    0         0         0         1.1111
```

第二种格式：　　[L, U, P] = lu(A)

紧接上问，在同一 MATLAB Command Window 中键入语句：

```
[L,U,P] = lu(A);
X = inv(U) * inv(L) * P * b
```

结果为：

```
X = -4.4000 5.4000 -6.6000 4.6000
```

下三角矩阵 L：

```
L = 1.0000    0         0         0
    0.5000    1.0000    0         0
    0.5000    0.1667    1.0000    0
    1.0000    0        -0.6667    1.0000
```

上三角矩阵 U：

```
U  = 2.0000     4.0000     6.0000     8.0000
     0          6.0000     3.0000    -1.0000
     0          0          1.5000     3.1667
     0          0          0          1.1111
```

交换矩阵 P:

```
P  = 1     0     0     0
     0     0     0     1
     0     1     0     0
     0     0     1     0
```

L 和 U 的乘积:L * U

```
ans  = 2     4     6     8
       1     8     6     3
       1     3     5     7
       2     4     5     7
```

P 的逆和 LU 的乘积:inv(P) * L * U

```
ans  = 2     4     6     8
       1     3     5     7
       2     4     5     7
       1     8     6     3
```

小结:两种格式都可以得到正确的结果,但实现的方式有一定差别。使用第一种格式时,L 并不是真正的下三角矩阵,其上三角有非零元素,主对角元也不为 1,因此,这里得到的 L 和 U 矩阵并不是严格意义上的三角分解,如果不使用交换矩阵 P,其严格意义的三角分解是不存在的。采用第二种格式时,得到的 L 是真正的下三角矩阵,其上三角元素均为零,主对角元为 1。但 L 和 U 矩阵的乘积并不是原来的 A 矩阵,元素相同,行的排列顺序不同。由交换矩阵 P 的定义有:

$$PA = LU$$

$$A = P^{-1}LU$$

因此,该线性方程组的解也相应变为:

$$X = L^{-1}U^{-1}b$$

如果待分解的系数矩阵 A 是对称的正定矩阵,可以通过三角分解将其变换为下三角矩阵和其转置的乘积的形式:

$$A = DD^T = \begin{pmatrix} d_{11} & & & \\ d_{21} & d_{22} & & \\ \vdots & \vdots & \ddots & \\ d_{n1} & d_{n2} & \cdots & d_{nn} \end{pmatrix} \begin{pmatrix} d_{11} & d_{12} & \cdots & d_{n1} \\ & d_{22} & \cdots & d_{n2} \\ & & \ddots & \vdots \\ & & & d_{nn} \end{pmatrix}$$

此三角分解称为 Cholesky 分解。MATLAB 提供了函数 chol()实现这一功能,当然,熟悉线性代数的读者都清楚,只有正定矩阵才能作上述分解,非正定矩阵有负的特征值,其平方根为复数,不可能分解出实数的三角矩阵。函数的具体用法如下:

```
A =    2     1
       1     1
```

```
    chol(A)
ans = 1.4142    0.7071
      0         0.7071
```

1.3.3 矩阵的特征值

矩阵的特征值这个概念在控制理论研究和大规模数据处理领域有着非常重要的意义。对于线性系统来说,判断其系统稳定性的最直接的办法就是根据其特征多项式的根,求解系统的极点,然后根据极点的分布情况给出结果:极点均在 s 域的左半平面的系统是稳定的,反之则为不稳定。而系统特征多项式的根实际上就是其状态空间表示下系数矩阵的特征值。下面简要回顾一下矩阵的特征值及其相关概念,并通过一个实例给出具体的应用。

在数域 P 内,对于一个 $n\times n$ 的矩阵 A,如果存在标量 λ 和非零向量 x 满足下述等式:

$$Ax=\lambda x$$

则标量 λ 称作矩阵 A 的特征值,向量 x 称作矩阵 A 的特征向量。对于同一个特征值,满足上述等式的特征向量有无穷多个,它们相差常数倍。如果 A 是满秩的矩阵,那么它拥有非零的 n 个特征值。如果这 n 个特征值互不相同,那么其特征向量相互间线性独立,组成的矩阵也是满秩的。如果满秩的 A 矩阵有 m 个($m\leqslant n$)相同的特征值,那么其特征多项式有 m 重根,该特征值对应的特征向量的解空间也是 m 维的。如果 A 的秩小于 n,那么其特征向量里有零元素。MATLAB 提供了一条求特征值的函数 eig(),可以方便的求出方阵 A 的特征和特征向量。

对于病态矩阵,在求解特征值和特征向量时有一个经常用到的概念叫做条件数,这是表征矩阵对其元素变化敏感程度的物理量,其定义是该矩阵的最大奇异值与最小奇异值之比。所谓奇异值就是矩阵 A 与其转置之积的特征值的平方根,其中 A 是任意矩阵,可以是方阵或长方阵。一般说来,矩阵的条件数越大,对元素变化的敏感程度越高,用这样的矩阵求解线性方程组或分析系统特性得到的结果越不可靠。尤其是在求解矩阵特征值时,甚至元素的舍入误差都会对所得的结果造成很大影响。然而在解决实际问题时,无论是数据采集还是计算机的量化过程中,矩阵元素的数值误差是不可避免的,这就需要寻找另外的解决途径。常用的做法是对矩阵 A 引入均衡变换:

$$B=T^{-1}AT$$

其中 T 是对角矩阵,其目的是保证矩阵 B 的行和列有基本相似的范数。一般通过均衡变换得到的矩阵 B 都有较低的条件数。MATLAB 提供了求方阵 A 条件数的函数 cond()和对其进行均衡变换的函数 balance(),其中对角变换阵 T 的元素均为 2 的整数倍,这样可以保证在计算时不会引入舍入误差。MATLAB 还提供了一些与求矩阵特征值相关的函数,如求矩阵的秩 rank(),求矩阵的行列式 det()等等,通过下面的例子,读者可以熟悉以上这些函数的具体用法。

考虑如下矩阵 A:

$$\begin{pmatrix} 1 & 10 & 100 & 100000 \\ 0.001 & 1 & 100 & 10000 \\ 0.00001 & 0.1 & 1 & 100 \\ 0.1 & 10 & 1000 & 10000 \end{pmatrix}$$

[例 1.3] 试求其行列式、阶数、特征向量和特征值,并分析其对元素变化的敏感程度,是否需要均衡变换。

分析:该矩阵的元素之间数值相差很大,是典型的病态矩阵,并且有可能不满秩。如果直接用这样的矩阵进行运算,所得的结果在数值上很不稳定,甚至会得出完全错误的结果。因此,为了保证计算结果的有效性,一般需要对原矩阵进行变换矩阵为初等矩阵的相似变换,使变换后的矩阵元素数值相差不大,行和列之间的范数值也比较接近,从而具有较好的鲁棒性。

求解过程:

在 MATLAB Command Window 中键入下述语句:

```
A = [1,10,1000,100000;0.001,1,100,10000;0.00001,0.01,1,100;0.1,10,1000,10000];
cond(A);
det(A);
rank(A);
norm(A(:,1))
norm(A(:,2))
norm(A(:,3))
norm(A(:,4))
[V,D] = eig(A);
cond(V);
```

结果为:

```
条件数:cond(A) = 2.9611e+019
行列式:det(A) =  2.2554e-011
阶  数:rank(A) =  3
矩阵 A 的列范数: norm(A(:,1)) =  1.0050
               norm(A(:,2)) =  14.1775
               norm(A(:,3)) =  1.4177e+003
               norm(A(:,4)) =  1.0100e+005
特征值:D  = 1.0e+004  *
    1.0021        0           0          0
    0          -0.0019        0          0
    0             0         0.0001       0
    0             0           0        0.0000
特征向量:V  =  -0.9901     0.9945     1.0000     -0.0000
              -0.0990     0.1047    -0.0052     -1.0000
              -0.0010     0.0010    -0.0001      0.0100
              -0.0992    -0.0002     0.0000     -0.0000
```

特征向量的条件数 cond(V) = 1.1127e+003

可以看出原矩阵 A 的条件数非常大,且阶数为 3,行列式接近于零(MATLAB 推荐使用条件数来判断是否为奇异矩阵),其各列的范数值相差也很大。并且,其特征向量所组成矩阵的条件数也非常大,不适于进行后续运算,需要对其进行均衡变换:

紧接上一问,在 MATLAB Command Window 中键入下述语句:

```
[T,B] = balance(A);
[V1,D1] = eig(B);
cond(V1);
norm(B(:,1));
norm(B(:,2));
norm(B(:,3));
norm(B(:,4));
```

结果为:

变换矩阵:T = 1.0e+003 *

1.0240	0	0	0
0	0.0320	0	0
0	0	0.0003	0
0	0	0	0.0010

变换后矩阵:B = 1.0e+004 *

0.0001	0.0000	0.0000	0.0098
0.0000	0.0001	0.0001	0.0313
0.0000	0.0001	0.0001	0.0400
0.0102	0.0320	0.0250	1.0000

B 矩阵的特征值:D1 = 1.0e+004 *

1.0021	0	0	0
0	-0.0019	0	0
0	0	0.0001	0
0	0	0	0.0000

B 矩阵的特征向量:V1 =

-0.0097	0.1797	0.9650	-0.0000
-0.0311	0.6051	-0.1614	-0.6156
-0.0399	0.7745	-0.2066	0.7880
-0.9987	-0.0405	0.0004	-0.0000

特征向量的条件数:cond(V1) = 1.2959

B 矩阵的列范数 norm(B(:,1)) = 102.4049

norm(B(:,2)) = 320.0043

norm(B(:,3)) = 250.0033

norm(B(:,4)) = 1.0013e+004

小结:考察变换后的矩阵 B,其列范数相对于原矩阵 A 来说,是比较接近的。因此,其特征向量组成矩阵 V1 的条件数也就小了很多,可以认为不再是病态矩阵了。我们还可以看到,B 矩阵的特征值与原矩阵 A 的特征值是一样的,这是因为矩阵的初等变换不改变原矩阵的特征值。事实上,这也是均衡变换的基础,否则就失去意义了。但需要注意的是,矩阵 B 的特征向量并不是原矩阵 A 的特征向量,而是变换后的结果。如前文所述,同一特征值的特征向量有无穷多个,它们组成一维(无相同特征值)或 m 维(m 个相同特征值)解空间,MATLAB 选取的是此解空间中二范数值为 1 的特征向量。

有了特征向量组成的矩阵 V,就能将原矩阵 A 化为对角矩阵的形式:

$$A = V^{-1}AV$$

即:

$$\begin{pmatrix} \lambda_{11} & & & \\ & \lambda_{22} & & \\ & & \ddots & \\ & & & \lambda_{nn} \end{pmatrix} = \begin{pmatrix} v_{11} & v_{12} & \cdots & v_{1n} \\ v_{21} & v_{22} & \cdots & v_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ v_{n1} & v_{n2} & \cdots & v_{nn} \end{pmatrix}^{-1} \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{pmatrix} \begin{pmatrix} v_{11} & v_{12} & \cdots & v_{1n} \\ v_{21} & v_{22} & \cdots & v_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ v_{n1} & v_{n2} & \cdots & v_{nn} \end{pmatrix}$$

但熟悉线性代数的读者都清楚，在解空间中并不是每个 $n \times n$ 的矩阵都能化为上述的对角形式。有些矩阵只能化为由对角矩阵组成的约当标准型。对于控制领域和信号处理中经常遇到的 $m \times n(m \neq n)$ 型长方阵，上述 eig()函数也是无能为力的。为此，MATLAB 专门提供了解决上述问题的函数，如求解约当标准型的 jordan()，求解长方阵奇异值的 svd()函数等等。这些函数在求解控制系统规范型、设计多变量解耦控制器和滤波器系数设计等方面有广泛的应用。

1.3.4　特殊矩阵

在控制系统设计领域要用到许多特殊形式的矩阵，例如求解传递函数最小实现时用到的 Hankel 矩阵，系统参数和阶次辨识时用到的随机矩阵等等。MATLAB 提供了许多生成和操作这些特殊矩阵的函数，并且都是考虑到各种极端的情况，经过专业人员测试通过的，可以放心的使用。下面有选择的简要介绍几种重点的特殊几种函数。

1. 单位矩阵

单位矩阵可以说是用途最广泛的矩阵之一，MATLAB 提供了一条 eye()函数可以用来生成指定维数的单位矩阵。这条函数有两种调用格式，第一种是 eye(n)，生成 n 维的单位矩阵；第二种是 eye(m,n)生成一个 $m \times n$ 维的矩阵，其主对角元为 1，其余元素为零。

2. 对角矩阵

所谓对角矩阵就是主对角元为非零元素，其余元素均为零的方阵，在数学参考文献中一般记为：

$$\mathrm{diag}(\alpha_1, \alpha_2, \cdots, \alpha_n) = \begin{pmatrix} \alpha_1 & & & \\ & \alpha_2 & & \\ & & \ddots & \\ & & & \alpha_n \end{pmatrix}$$

MATLAB 提供了一条函数 diag()，可以用来生成对角矩阵。其调用格式为 diag(V, K)，V 是维数为 n 的向量，K 是标量。如果省略 K 或 K 为零，则生成对角元素为 V 的对角方阵；如果 K 不为零，那么生成秩为 n 加 K 的绝对值的方阵，如果 K 大于零则从主对角元向上数第 K 重对角元为 V，反之则向下数。例如：

```
m = 1;
diag(-m:m) + diag(ones(2*m,1),1) + diag(ones(2*m,1),-1)
```

其结果为：

```
ans = -1     1     0
       1     0     1
       0     1     1
```

3. 随机矩阵

所谓随机矩阵就是矩阵的各个元素是随机生成的。MATLAB 的 rand()函数生成的

随机矩阵的元素满足[0,1]区间上的均匀分布。其格式为 rand(m,n),m 和 n 分别为随机矩阵的行数和列数。

熟悉仿真算法的读者清楚,由计算机生成的随机数只是伪随机数,为了降低数据的相关性应该随时更换随机数发生器的种子数。在 MATLAB 中,这种想法可以通过调用该函数的 rand('seed',s)格式来实现。其中 s 是标量,可以通过不同的 s 值来改变机器随机数发生器的种子数,s = 0 则设回缺省值。如:

```
rand(1,4)
ans = 0.2190 0.0470 0.6789 0.6793
rand('seed',1)
rand(1,4)
ans = 0.5129 0.4605 0.3504 0.0950
rand('seed',0)
rand(1,4)
ans = 0.2190 0.0470 0.6789 0.6793
```

如果希望得到正态分布的随机矩阵,则应该调用 randn()函数,其格式与 rand()函数基本类似,缺省是均值为 0,方差为 1 的正态分布的随机数。

4. Vandermonde 矩阵

将控制系统的可控规范型化为特征值规范型时,所用到的变换矩阵就是由原规范型的系数矩阵特征值组成的范德蒙(Vandermonde)矩阵。控制领域使用的 Vandermonde 矩阵的具体形式为:

$$W = \begin{pmatrix} 1 & 1 & \cdots & 1 \\ \lambda_1 & \lambda_2 & \cdots & \lambda_n \\ \lambda_1^2 & \lambda_2^2 & \cdots & \lambda_n^2 \\ \vdots & \vdots & \vdots & \vdots \\ \lambda_1^{n-1} & \lambda_2^{n-1} & \cdots & \lambda_n^{n-1} \end{pmatrix}$$

MATLAB 提供了一条实现此 Vandermonde 矩阵的函数 vander(),但是使用此函数生成的 Vandermonde 矩阵和上述的 W 矩阵相差 90°。也就是说,将用 vander()函数生成的矩阵元素逆时针旋转 90°后就是控制领域常用的 Vandermonde 矩阵。MATLAB 提供了一条函数 rot90()专门用来对矩阵的元素进行逆时针旋转 90°的操作,请看下例:

```
C = [1 2 3 4];
vander(C)
% Vandermonde 矩阵及其旋转结果
ans = 1     1     1     1
      8     4     2     1
      27    9     3     1
      64    16    4     1
rot90(ans)
ans = 1     1     1     1
      1     2     3     4
      1     4     9     16
```

```
     1     8    27    64
```

5. Hankel 矩阵

在判断系统最小实现的维数时使用的 Hankel 矩阵是由 Markov 参数矩阵生成的，和 MATLAB 提供的 hankel()函数生成的矩阵略有不同，但其具体的形式是一样的。下面按照 MATLAB 格式进行叙述。所谓 Hankel 矩阵，一般是指形如

$$H=\begin{pmatrix} h_1 & h_2 & \cdots & h_n \\ h_2 & h_3 & \cdots & h_{n+1} \\ \vdots & \vdots & \ddots & \vdots \\ h_n & h_{n+1} & \cdots & h_{2n-1} \end{pmatrix}$$

的方阵，它是对称矩阵，且反对角线的各元素均相等。MATLAB 提供了两种格式来生成 Hankel 矩阵，一种是 hankel(C)，生成上述标准的 Hankel 矩阵，C 是该 Hankel 矩阵的第一列向量，其余反对角线以下的元素均为零；如果不希望 Hankel 矩阵的反对角线以下元素为零，可以采用另一种格式，即 hankel(C,R)，其中 C 和 R 均为 n 维向量，各生成一个标准 Hankel 矩阵的上半部分，然后拼接起来，得到一个 $n\times n$ 维的 Hankel 矩阵，需要注意的是 C 的最后一个元素和 R 的第一个元素应该相等，MATLAB 用这个相等的元素作为该 Hankel 矩阵的反对角线元素，否则就会提出警告，直接用 C 下例的最后一个元素作为反对角线元素。请看下例：

```
C=[1 2 3 4];
R=[4 5 6 7];
%标准的 Hankel 矩阵
hankel(C)
ans =   1    2    3    4
        2    3    4    0
        3    4    0    0
        4    0    0    0
%合成的 Hankel 矩阵
hankel(C,R)
ans =   1    2    3    4
        2    3    4    5
        3    4    5    6
        4    5    6    7
```

1.3.5 矩阵的非线性运算

前边介绍的主要是加、减、乘、除等矩阵的基本运算，包括求逆、对角化、旋转等线性变换在内，这些运算统称为矩阵的线性运算。而矩阵的非线性运算，一般的读者在学习线性代数的过程中恐怕接触得比较少。这些非线性运算主要是根据不同的工程需要发展起来的，一方面是实际运算的需要，另一方面也是为了书写简洁。这些非线性运算主要分为两大类：一类是对矩阵元素的非线性运算，基本上标量的非线性运算都可以在矩阵运算中找到对应的版本（如前所述，事实上 MATLAB 运算的基本元素是矩阵，标量的运算也是以矩阵的方式进行的）；另一类是对于整个矩阵的非线性运算，包括求取矩阵的指数、对数或

者是任意表达式的函数等等，事实上这些对整个矩阵的非线性运算都是用加、减、乘、除等线性运算来近似的，但使用 MATLAB 提供的矩阵非线性运算函数，可以事半功倍，省去了高级语言编程的工作量，迅速得到相的结果。

1. 矩阵元素的非线性运算

矩阵元素的非线性运算是对矩阵的单个元素进行的。当然，用户即可以对矩阵中指定的元素进行非线性运算，也可以对矩阵中每个元素整体进行同样的非线性运算。有关标量的非线性运算基本都可以对矩阵进行，只是需要特别注意该非线性函数的实现条件。

表 1-1 列出了一些常用的矩阵非线性函数，需要更多的信息请查阅 MATLAB 函数手册或帮助文件。

表 1-1 MATLAB 常用矩阵元素的非信息运算

数值函数		复数函数	
Fix, floor, ceil, round	取整运算	conj, image, real	复数共轭、虚部、实部
Rem	除后余数	abs	绝对值
Sign	符号函数	angle	相角
指数函数		**三角函数**	
exp	指数	sin, cos	正、余弦
log	自然对数	asin, acos	反正、余弦
log10	常用对数	sinh, cosh	双曲正、余弦
sqrt	平方根	asinh, acosh	反双曲正、余弦

这些函数的调用格式也和标量函数相同，请看下例：

```
A=[1 2 3 4;5 6 7 8;4 3 2 1;8 7 6 5]
A =  1    2    3    4
     5    6    7    8
     4    3    2    1
     8    7    6    5
log(A)
ans =  0        0.6931   1.0986   1.3863
       1.6094   1.7918   1.9459   2.0794
       1.3863   1.0986   0.6931   0
       2.0794   1.9459   1.7918   1.6094
log(A(:,1)
ans =    0
         1.6094
         1.3863
         2.0794
```

其中 A(:,i)表示 A 矩阵的第 i 列元素。

2. 整个矩阵的非线性运算

MATLAB 提供了一条函数 funm()可以进行矩阵一般函数运算，不过 MATLAB 在其帮助文件中指出，这条函数语句在实现过程中使用了一种潜在不稳定的算法，尤其是对于

病态矩阵,有可能得出不正确的结果。如果该函数内嵌的检查器检查出不符合计算条件的矩阵,MATLAB 就会自动报警。但这个检查器过于灵敏,虚警概率较高。因此,对于常用的矩阵指数和对数等运算,MATLAB 推荐使用其专有函数 expm()、logm(),这些函数的实现算法与 funm()不同。

funm()的调用格式为:F = funm(A,'fun'),一般要求 A 是方阵。我们仍然使用上例的矩阵 A,有:

```
funm(A,'cos')
WARNING: Result from FUNM may be inaccurate. esterr = 1
ans =        0.2541     -0.3325      0.0809      0.4943
            -0.8214      0.5920      0.0054      0.4189
             0.6515      0.2381      0.8247     -0.5887
             0.5760      0.1626     -0.2508      0.3358
```

可以看到 MATLAB 警告说 funm()函数计算结果可能不准确。为了保险起见,对于一般的非线性运算可以用 Taylor 级数来近似,用 MATLAB 语言编程只需很短的程序就可以实现,精度和准确性都可以人为设定。我们将在 MATLAB 编程这一节中具体介绍编程实现矩阵非线性运算的方法。

在控制系统设计和仿真领域,使用最多的矩阵非线性运算就是矩阵指数运算,其定义为:对于 $n\times n$ 的方阵 A,有:

$$e^{At}=I_n+At+\frac{1}{2!}A^2t^2+\cdots+\frac{1}{k!}A^kt^k+\cdots=\sum_{k=0}^{\infty}\frac{A^kt^k}{k!}$$

对于控制系统的状态空间描述形式 $Ax=bu$,因为下式的存在:

$$L[e^{At}]=(sI-A)^{-1}$$

即状态空间系数矩阵 A 的指数的拉普拉斯变换等于其预解矩阵,而预解矩阵是求解状态空间方程的关键,因此,指数矩阵获得了非常广泛的应用。状态方程的求解、系统的模态判定等等都要用到指数矩阵的概念。

MATLAB 提供了四条函数可以求解矩阵指数,分别是 expm()、expm1()、expm2()、expm3。其中 expm()和 expm1()都使用 Pade 算法来逼近矩阵指数的真实值,对于没有特殊限制的矩阵,一般推荐使用这两个函数,能够保证算法的收敛性,其缺点是速度比较慢。这两个函数的区别是:expm()嵌在系统内核里,而 expm1()是以 M 文件的格式存为外部函数。expm2()使用上述矩阵指数的定义式来计算,即 Taylor 展开法。这种方法的优点是结构和思路都比较简单,适于用户自己修改和控制,满足其特殊要求。其缺点是速度比较慢且不能保证算法一定收敛(绝大部分情况是可以保证得到正确结果的)。expm3()对特征向量满秩的矩阵进行了优化,通过 A 矩阵的特征向量和特征值来计算其指数矩阵,算法的核心部分为:

```
[V,D] = eig(X)
expm3(X) = V * diag(exp(diag(D)))/V
```

因此,其算法的准确性取决于 A 矩阵的特征向量矩阵的具体条件。如果 A 矩阵的特征值有重根的话,其特征向量组成的矩阵行列式接近于零,在求逆的过程中可能会得出不正确的结果,进而影响最终的准确度。请看下例:

[例 1.4] 考虑如下矩阵 A,试比较分别用 expm()、expm2()和 expm3()求解其矩阵

指数的结果,并讨论其原因。

$$A = \begin{pmatrix} 2 & 3 & 4 & 5 \\ 0 & 2 & 3 & 4 \\ 0 & 0 & 4 & 5 \\ 0 & 0 & 0 & 4 \end{pmatrix}$$

分析:明显可以看出,矩阵 A 有两个二重特征值 2 和 4,在使用特征向量法求解矩阵指数时有可能得到不正确的结果。

求解过程:

在 MATLAB Command Window 中键入下述语句:

```
A=[2 3 4 5;0 2 3 4;0 0 4 5;0 0 0 4]
%采用 expm()函数
expm(A)
ans =   7.3891     22.1672    167.3879    691.5055
        0           7.3891     70.8136    326.8702
        0           0          54.5982    272.9908
        0           0           0          54.5982
expm2(A)
ans =   7.3891     22.1672    167.3879    691.5055
        0           7.3891     70.8136    326.8702
        0           0          54.5982    272.9908
        0           0           0          54.5982
%与矩阵 A 相同
logm(ans)
ans =     2.0000     3.0000     4.0000     5.0000
         -0.0000     2.0000     3.0000     4.0000
         -0.0000    -0.0000     4.0000     5.0000
         -0.0000    -0.0000     0.0000     4.0000
%与 expm( )函数的结果不同
%采用 expm3( )函数
expm3(A)
%系统警告
Warning: Matrix is close to singular or badly scaled.
         Results may be inaccurate. RCOND = 1.169706e-018.
> In D:\MATLAB\toolbox\matlab\matfun\expm3.m at line 13
%求解结果
ans =   7.3891        0        200.6386         0
        0           7.3891      70.8136         0
        0             0         54.5982         0
        0             0           0           54.5982
%与矩阵 A 不同
logm(ans)
ans =   2.0000        0          8.5000         0
```

```
        0           2.0000     3.0000         0
        0              0       4.0000         0
        0              0          0        4.0000
%矩阵 A 的特征值和特征向量
[V,D] = eig(A)
%接近奇异
V =   1.0000     -1.0000     0.9206     -0.9206
        0         0.0000     0.3249     -0.3249
        0            0       0.2166     -0.2166
        0            0          0        0.0000
D =     2        0        0        0
        0        2        0        0
        0        0        4        0
        0        0        0        4
inv(V)
%系统警告
Warning: Matrix is close to singular or badly scaled.
         Results may be inaccurate. RCOND = 6.599346e-019.
%求解结果
ans = 1.0e+017 *
         0.0000      1.1111      -1.6667      0
         0           1.1111      -1.6667      0
         0           0            0.0000      8.5492
         0           0            0           8.5492
%手工计算的 expm3()函数
%expm3()函数的核心算法
E = V * diag(exp(diag(D)))/V
%系统警告
Warning: Matrix is close to singular or badly scaled.
         Results may be inaccurate. RCOND = 1.169706e-018.
%求解结果
E =   7.3891      0          200.6386     0
        0         7.3891      70.8136     0
        0         0           54.5982     0
        0         0            0          54.5982
```

小结:从以上的代码和运行结果中可以看出,expm3()函数在求解有重特征值矩阵的指数时有可能得到不正确的结果,而使用 expm()和 expm2()时一般会得到正确的结果。分析并运行了 expm3()函数的核心代码,发现主要原因是其中用到了特征向量矩阵求逆这一步,对于不满秩的特征向量矩阵是无法求出正确结果的。对于类似这种不满秩矩阵求逆的运算,MATLAB 并不中途退出,而是继续运行,并提出警告"Results may be inaccurate. RCOND = 1.169706e-018.",RCOND 是该矩阵条件数的倒数,接近零表示矩阵为病态条件,可能会得出不正确的结果。虽然 MATLAB 对于这类错误的查错能力非常强,

但我们建议,如果不是对计算时间要求过高的话,一般使用 expm()或原始的定义式 expm2()。

1.4 MATLAB 绘图

MATLAB 的绘图功能和它的矩阵计算能力同样出色,它所推崇的科学可视化(Scientific Visualization)概念,把科学工作者从高级语言烦琐的图形处理算法中解脱出来,图形操作变得像搭积木一样简单。如果使用高级语言进行绘图,首先要调用图形库,进行屏幕初始化,这需要绘图者对所使用计算机的硬件有一定了解;然后再调用需要的库函数,设置所绘制图形的具体参数,这需要绘图者事先就要对所绘制的图形有大致了解,否则就必须不断地修改程序,调整到满意为止。

在 MATLAB 环境下,上述这一切都不需要去了解,使用者只需知道几条简单的绘图函数就可以绘制出精美的二维或三维直角坐标图形。并且,MATLAB 还能绘制直方图、阶梯图、向量场图、极坐标图等特殊要求的二维图形,如果用高级语言来解决这些问题的话通常要耗费大量精力,还未必能得到满意的结果。

至于三维图形的绘制,如果不是专业设计人员的话,仅仅使用高级语言是很难做到令人满意的程度。这需要绘图者有较强的空间想像能力和丰富的绘图经验。而在 MATLAB 环境下,对于使用者来说,三维图形绘制与二维图形绘制在编程方面没有什么区别,思路和实现方式都基本类似。

使用 MATLAB 绘图另一个吸引人的地方就是与使用机器的无关性,无论是 PC 机还是 Alpha 工作站上使用的 MATLAB 都采用同样格式的绘图函数,使用者不必去关心所使用机器的具体的硬件细节,只需掌握基本的操作命令就可以轻松地将需要的程序移植到不同的计算机上。

对于控制领域的科技人员来说,MATLAB 强大的绘图功能的确是非常令人激动的。它使得用图形来验证理论结果不再是一件烦琐不堪的事情。事实上,这也是 MATLAB 一问世就受到控制界广泛关注的重要原因之一。

早期的控制系统计算机辅助设计软件最大的缺憾之一就是不能提供图形绘制的接口,使得具体应用起来不直观,只能通过具体的数据间接地分析系统的性能。MATLAB 非常漂亮地解决了这一问题,在 Simulink 中甚至可以直接输出系统的响应曲线,不必通过命令窗口。

控制工程师可以非常方便地应用 MATLAB 的绘图功能进行系统仿真,对理论计算进行直观的检验。尤其是在过程控制领域,工程师可以根据系统的响应曲线来判定参数设定的优劣,不必再根据性能指标作间接的估计。通过系统的开环响应曲线,还可以指导系统 PID 参数的修改。即使是控制专业的学生,也可以通过 MATLAB 的绘图功能对所接触的问题有一个感性的认识,巩固学习效果。

1.4.1 二维直角坐标图形绘制

读者在上一节的最小二乘拟合的例子中,已经接触到了 MATLAB 最基本的二维图形绘制语句 plot(),这是绘制线性 x－y 坐标图最常用的命令。使用这条语句绘制图形时

不必考虑坐标平面的大小，x 轴和 y 轴的刻度也是自动标出的。我们首先通过下面这个简单的例子，熟悉 plot()语句的用法：

［例 1.5］　在 MATLAB Command Window 中键入下述语句：

```
% Line plot of a chirp
x = 0:0.05:5;
y = sin(x.^2);
plot(x,y);
```

结果如图 1－5 所示。

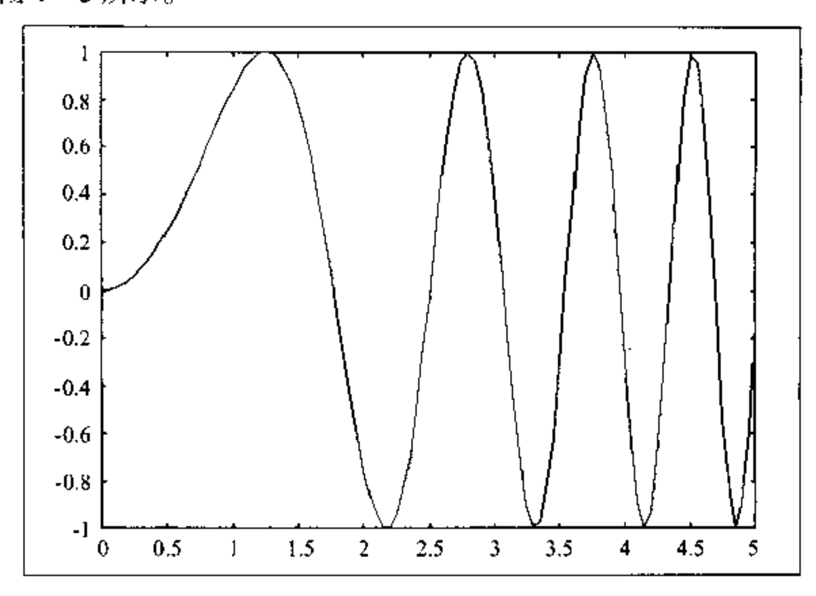


图 1－5　plot()语句基本绘图格式

事实上，plot()语句提供了繁多的可选参数和修饰功能，其完整的调用形式如下：

PLOT(X1,Y1,S1,X2,Y2,S2,X3,Y3,S3,...)

其中 Xi 表示 X 轴的数据，Yi 表示 Y 轴的数据，Si 是可选的控制参数，控制输出图形的颜色或线条的类型。其可选参数见表 1－2：

表 1－2　plot()函数可选参数列表

参数名	功　能	参数名	功　能	参数名	功　能
y	黄色	p	五角星形	v	下三角形
m	深紫色	q	六边形	^	上三角形
c	蓝绿色	.	点	<	左三角形
r	红色	x	x 形	>	右三角形
g	绿色	+	+ 号	-	实线
b	蓝色	*	* 号	:	点线
w	白色	s	正方形	-.	点划线
k	黑色	d	钻石形	--	虚线

不同类型的可选参数是可以叠加使用的，例如 PLOT(X,Y,'c+:')表示以"+"号取代"."号画一条蓝绿色的+号线。绘图时还可以为图形增加注释、题头或网格，常用的相关语句有：title，画题头；xlabel，x 轴标注；ylabel，y 轴标注；text 标注数据点；grid 加网格。使用这些可选参数和辅助语句稍加修饰之后，MATLAB 就可以绘制出精美的二维图形，满足各种不同的需要。请看下例：

在同一坐标图下绘制出函数 $y=\sin(x)$，$x\in[-4\pi,4\pi]$ 和函数 $z=\sin(x)/x$，$x\in[-4\pi,4\pi]$，要求用不同的线段类型表示，并加标注和网格。

结果：如图 1-6 所示。

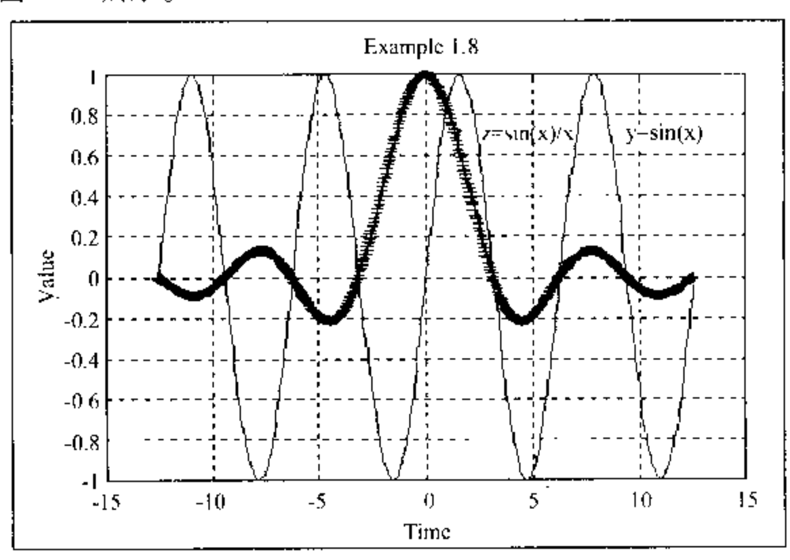


图 1-6 plot()语句完整格式调用

分析：plot()语句中的参数是以向量格式存储的，对于这种没有给出具体数据，而只给定函数解析式的图形绘制问题，一般选取 x 轴为时间轴，根据不同的精度要求，隔一定的距离(一般是很小的常数)选一个点，计算其函数值，然后描点连线，最后加标注。同时，为了提高数据的利用率，可以将所选的点以及其函数值存在不同的向量中，方便 plot()语句调用。

求解过程：

在 MATLAB Command Window 中键入下述语句：

```
%关闭所有图形窗并清屏
close all;
clf;
%数据初始化
x = - 4 * pi:0.05:4 * pi;
y = sin(x);
z = sin(x)./x;
%绘图部分
plot(x,y,'r-',x,z,'b+')
%图形注释
title('Example 1.8');
```

```
xlabel('time');
ylabel('value');
%在指定坐标处输出文字
text(1.5,0.7,'z=sin(x)/x');
text(9,0.7,'y=sin(x)');
grid;
```

键入回车后执行即可得到图 1-6 的结果。

小结:

plot()语句可以完成给定数据或函数的二维图形绘制的绝大部分工作,在使用过程中读者可以自己摸索可选参数的各种组合形式,这里只简单的选择了"r-"和"b+"两种类似的组合方式。

text()语句可以在输出图形的指定位置书写文字。当然,在事先清楚所绘制图形的形状和曲线分布的情况下,使用 text()语句可以很方便的在适当位置输出文字。如果对输出图形的形状不太熟悉的话可以先用 plot()语句画图,确定好位置后再修改 text()语句指定的参数位置。

如果读者不希望两条曲线出现在同一坐标系中,但仍要求不同的曲线出现在同一张图里,可以考虑使用 MATLAB 提供的 subplot()语句配合 plot()语句使用。该语句的基本调用格式为:

SUBPLOT(m,n,p)或 SUBPLOT(mnp)

其中 m、n、p 均为大于零的整数,其功能是将 MATLAB 的图形窗(Figure Window)均匀划分为 $m \times n$ 部分,并且分别建立 $m \times n$ 个坐标系,选中其中的第 p 个坐标系进行操作。首先用 subplot()语句对绘图窗口进行初始化,然后再使用 plot()语句,就可以在选中的坐标系中绘制图形了,而 plot()语句的其他可选参数和辅助语句都可以照常使用。请看下例:

将 MATLAB 绘制的图形窗分为四部分,设 $\omega t \in [0, 4\pi]$,以 0.02 为步长,分别绘制下列图形:

(1) 左上部分,ωt 为横坐标,绘制 $v = 120\sin\omega t$ 和 $i = 100\sin(\omega t - \pi/4)$。

(2) 右上部分,ωt 为横坐标,绘制 $P = vi$。

(3) 右下部分,ωt 为横坐标,绘制系统的一阶响应曲线。

$$y = 1 - e^{-\frac{\omega t - \tau}{T}}$$

其中: $\tau = -1, T = 2$

(4) 左下部分,ωt 为横坐标,绘制系统的二阶响应:

$$y = 1 - \frac{1}{\sqrt{1-\zeta^2}} e^{-\zeta\omega t}\sin(\omega_d \omega t + \theta)$$

其中 $\zeta = 0.4, \theta = \arctan\frac{\sqrt{1-\zeta^2}}{\zeta}, \omega_d = \sqrt{1-\zeta^2}$

结果:图 1-7 所示为 subplot()函数应用示例,四幅图形的横坐标同为时间,纵坐标是相应的各函数值或响应曲线。

分析:一般来说,MATLAB 图形窗的大小是一定的。因此,虽然 subplot()语句可以将图形窗分为任意 $m \times n$ 部分,但为了保证适当的清晰度,使用较多的是 1×2 或 2×2 的格式。

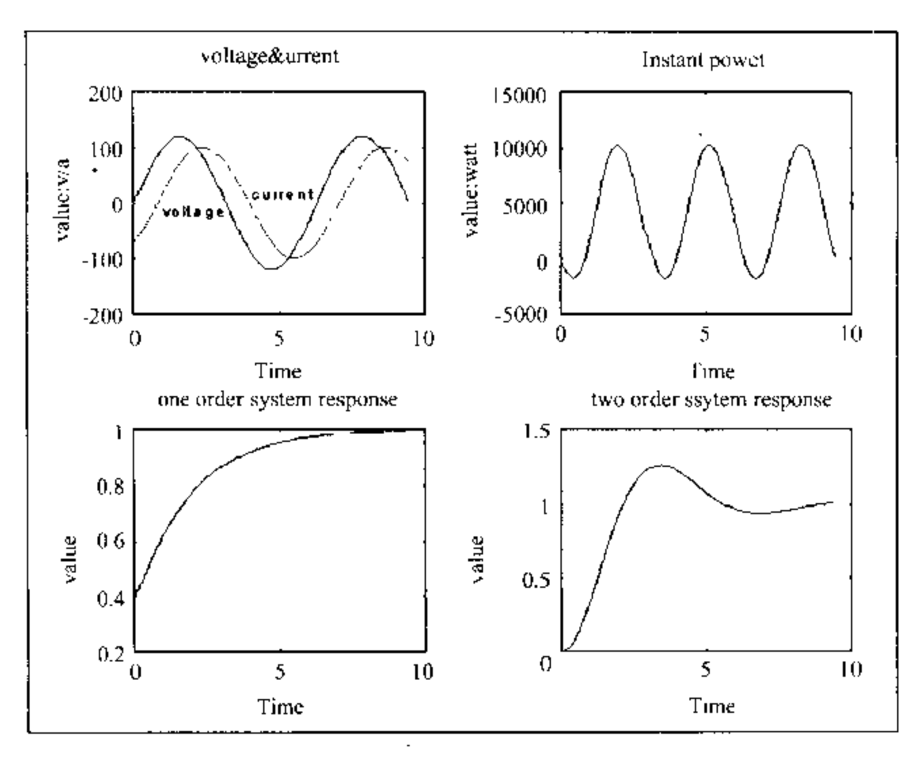


图 1-7 subplot()函数应用示例

求解过程：

在 MATLAB Command Window 中键入下述语句：

```
%关闭所有图形窗并清屏
close all;
clf;
%数据初始化
wt = 0:0.02:3 * pi;
%电压
v = 120 * sin(wt);
%电流
i = 100 * sin(wt - pi/4);
%瞬时功率
p = v. * i;
tao = - 1;
T = 2;
%一阶系统响应
y = 1 - exp( - (wt - tao)/T);
e = 0.4;
theta = atan(sqrt(1 - e * e)/e);
wd = sqrt(1 - e * e);
%二阶系统响应
yy = 1 - exp( - e * wt). * sin(wd * wt + theta)/sqrt(1 - e * e);
% 绘图部分
```

```
%左上部分
subplot(221),
plot(wt,v,wt,i);
title('voltage&current');
xlabel('time');
ylabel('value: v/a');
text(1,-20,'voltage');
text(4,10,'current');
grid;
%右上部分
subplot(222),
plot(wt,p);
title('instant power');
xlabel('time');
ylabel('value: watt');
grid;
%左下部分
subplot(223),
plot(wt,y);
title('one order system response');
xlabel('time');
ylabel('value');
grid;
%右下部分
subplot(224),
plot(wt,yy);
title('two order system response');
xlabel('time');
ylabel('value');
grid;
```

小结:本例再一次复习了 plot()语句的基本用法,并且介绍了 subplot()语句的基本调用格式。掌握了这两条基本语句之后,一般数学分析和控制系统仿真中的图形绘制问题都能够比较圆满地解决了。因为和数值分析类似,所有非线性变换都是以线性变换为基础的,可以用线性变换来近似。

同样,对于有特殊要求的坐标系的图形绘制问题,也可以通过坐标变换化为线性直角坐标,然后再用 plot()语句绘制。当然,MATLAB 中有一些直接绘制特殊坐标图形的语句,不过概念上是类似的。本例的第三、四部分的图是尝试用手工绘制一阶和二阶系统的阶跃响应曲线,一方面是熟悉复杂表达式的 MATLAB 实现,另一方面是逐步介绍一些控制系统的基本概念,从图形的实际效果来看还是比较令人满意的。事实上,MATLAB 的控制系统工具箱提供了,包括直接绘制系统阶跃响应曲线在内的一系列控制系统绘图函数,用户只需给出必要的参数就可由 MATLAB 自己直接绘制曲线,不必考虑图形的具体绘制问题,具体的内容将在以后的各章节中介绍。

本例的程序中有一条语句需要特别注意，即求瞬时功率的“p = v. * i”。因为 v 和 i 均为向量，如果直接使用“ * ”号就表示向量相乘。若是同维的行向量或列向量则表示作内积，否则 MATLAB 就会给出出错信息。“. * ”号则表示向量的对应元素相乘，其结果仍是同维的向量。同样，用到矩阵元素分别相除时，也要用“./”号。这是使用绘图语句的一个常用的小技巧。

相信读者阅读了上边的几个例子之后，一定对使用 MATLAB 进行二维图形绘制的优点有了初步的了解。事实上，在二维图形绘制领域，MATLAB 还有许多特殊的功能，包括专为金融领域和工程领域优化的柱形图、针状图、对数坐标图，以及各种领域都有广泛应用的极坐标图等等。下一节，我们将详细介绍与控制系统相关的 MATLAB 二维特殊坐标图形的绘制。

1.4.2 二维特殊坐标图形绘制

MATLAB 提供了许多特殊坐标的图形绘制语句，可以满足各种不同场合的需要。如统计和金融领域常用的柱形图，电子和控制领域常用的对数和半对数坐标图，以及复变函数和控制领域常用的极坐标图等等。经常用到的语句如表 1－3 所列：

表 1－3 MATLAB 常用特殊坐标绘图语句

语句	功能	语句	功能	语句	功能
bar	柱形图	hist	累计图	Fill	实心图
errorbar	图形加上误差范围	rose	极坐标累计图	Feather	羽毛图
fplot	较精确的函数图形	stairs	阶梯图	Loglog	双对数坐标图
polar	极坐标图	stem	针状图	Semilogx	半对数坐标图（x 轴对数）
compass	罗盘图	quiver	向量场图	Semilogy	半对数坐标图（y 轴对数）

这些语句的调用格式与 plot( )语句基本类似，同样也可以配合响应的可选参数或是 subplot( )语句使用。对于调用这些语句的具体格式在这里就不一一介绍了，下面通过一个简单的例子，使读者熟悉一下其基本用法，需要了解详细的情况请参阅相关的手册或是 MATLAB 帮助。

［例 1.6］ 选择合适的语句，在 MATLAB 图形窗中绘制以下函数的图形：

(1) 函数 $y = \sin(x)/x$ 的柱形图、阶梯图、针状图和实心图。

(2) 绘制并检验 2000 个高斯白噪声数据的概率分布。

(3) 绘制螺旋曲线 $r = 0.3e^{0.3\theta}$的极坐标图形。

结果如下：

第一问，函数 $y = \sin(x)/x$ 的柱形图、阶梯图、针状图及实心图，如图 1－8 所示；

第二问，2000 个高斯白噪声数据的概率分布图，如图 1－9 所示；

第三问，螺旋曲线 $r = 0.3e^{0.3\theta}$的极坐标图，如图 1－10 所示；

分析：如前所述，所有的二维特殊坐标图形都可以用线性直角坐标来近似，如极坐标可以用 $x = r\cos\theta$、$y = r\sin\theta$ 来求出其直角坐标表达式，并直接绘图。但对于其他形式的特殊坐标图形，实现起来就比较烦琐。尤其是有关概率统计的绘图，首先需要对原始数据进

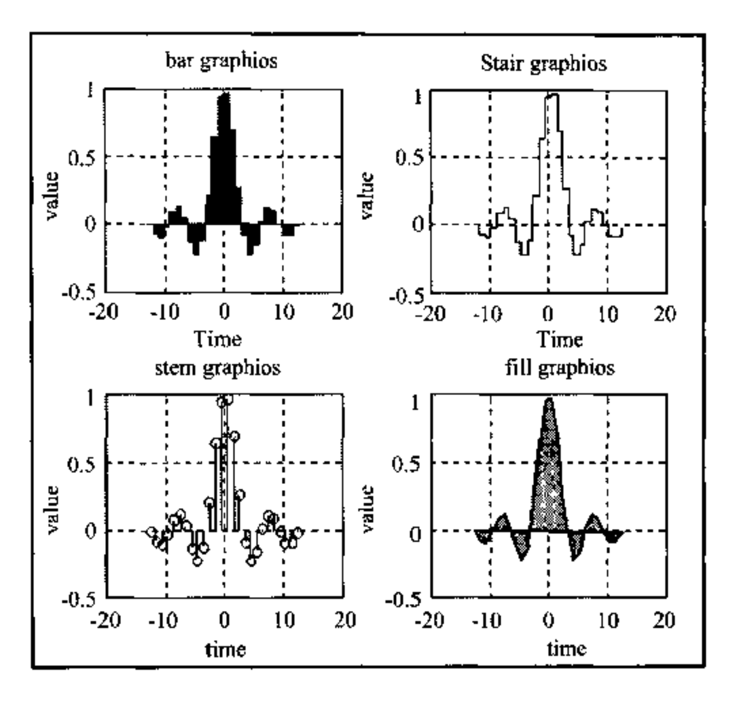


图 1-8 柱形图、阶梯图、针状图及实心图示

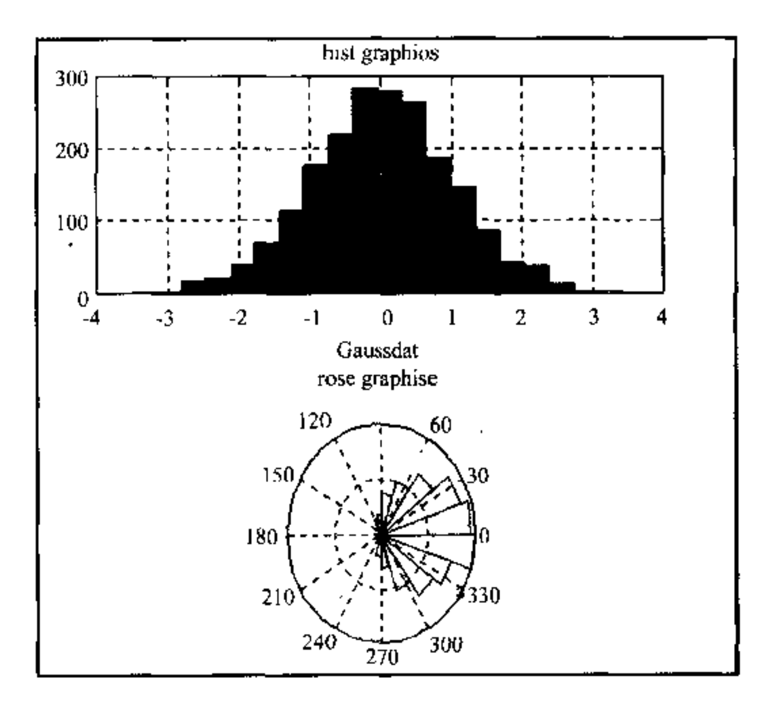


图 1-9 正态高斯白噪声概率分布图

行处理，统计出各区间的数据分布情况，然后在确定坐标范围进行绘制。笔者曾用 C 语言实现了上述统计正态高斯白噪声的程序，包括绘图大约需要 200 行左右，但用 MATLAB 实现只需要短短的几行。

求解过程：

在 MATLAB Command Window 中键入下述语句：

第一问，代码如下：

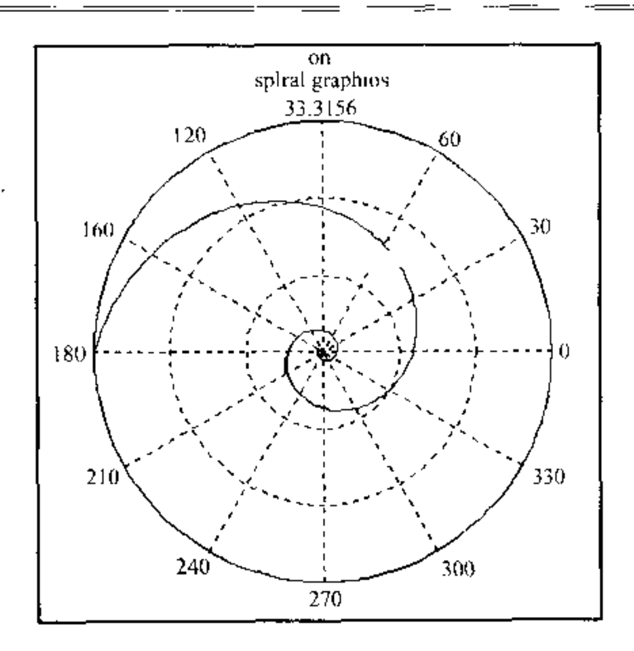


图 1-10 螺旋曲线极坐标图

```
%关闭所有图形窗并清屏
close all;
clf;
%数据初始化
x = -4 * pi:1:4 * pi;
y = sin(x)./x;
%绘图部分
subplot(221),
%柱形图
bar(x,y);
title('bar graphics');
xlabel('time');
ylabel('value');
grid;
subplot(222),
%阶梯图
stairs(x,y);
title('stair graphics');
xlabel('time');
ylabel('value');
grid;
subplot(223),
%针状图
stem(x,y);
title('stem graphics');
xlabel('time');
```

```
ylabel('value');
grid;
subplot(224),
%实心图
fill(x,y,'g');
title('fill graphics');
xlabel('time');
ylabel('value');
grid;
```

第二问,代码如下:

```
%关闭所有图形窗并清屏
close all;
clf;
%数据初始化,产生 2000 个正态分布的高斯白噪声
x = randn(2000, 1);
%绘图部分
subplot(211),
%直角坐标概率分布图
hist(x,20);
title('hist graphics');
xlabel('Gauss data');
ylabel('number');
grid;
subplot(212),
%极坐标概率分布图
rose(x);
title('rose graphics');
```

第三问,代码如下:

```
%关闭所有图形窗并清屏
close all;
clf;
%数据初始化
theta = 0:0.05:5 * pi;
r = 0.3 * exp(0.3 * theta);
%绘图部分
%极坐标图
polar(theta,r);
title('spiral graphics');
```

小结:通过上例,我们可以看出使用 MATLAB 绘制二维特殊坐标图形是多么得心应手。柱形图和阶梯图多用于金融和社会学领域,可以直观地表达出数据的变化趋势;针状图可以用来表达采样控制系统,其采样时间间隔可以通过时间坐标的间隔来控制;累计图和极坐标累计图是概率分析领域的得力助手,hist()和 rose()自动将需绘制的数据进行统

计处理，并按照概率分布图的形式分别在直角坐标和极坐标中绘制出来。在图 1－9 中我们可以看到非常标准的正态分布图，以及其极坐标形式。在进行系统辨识时，可以使用这两条以及对生成的高斯白噪声进行检验。

除了上述这些特殊坐标图形，在控制系统中还有一类常用的特殊坐标体系，即对数坐标体系。在评价系统的稳定性时常用的对数频率特性图（亦称 bode 图）使用的就是对数坐标轴。使用这种特别体系的好处一方面是能够把较大范围的变化情况集中体现在比较小的空间里，便于观察和评价；另一方面就是能够化乘、除、乘方运算为加、减与常数相乘的运算，使得整个系统的对数频率特性图由各子系统的特性图相加减就可以得到，大大地简化了工作量，其结果也具有指导意义。

在早期控制系统仿真手段不是很完善的情况下，绘制系统的对数频率特性图还是一种分析和设计控制系统的行之有效的方法。即使是现在，时域的研究理论和仿真工具非常先进了，对数频率特性图因为其简便、直观的特点还是拥有一定的生命力。MATLAB 提供了三条绘制对数坐标图形的语句：loglog()、semilogx()、semilogy()，分别可以绘制双对数坐标图、x 轴对数坐标图和 y 轴对数坐标图。能够非常方便地完成对数幅频特性曲线和对数相频曲线的绘制。请看下例：

考虑流程工业过程控制领域常用的一阶惯性系统，传递函数如下：

$$G = \frac{45}{j0.1\omega + 1}$$

试绘制其对数幅频特性曲线和对数相频曲线，并讨论系统性能。

结果：对数幅频特性曲线和对数相频曲线如图 1－11 所示。

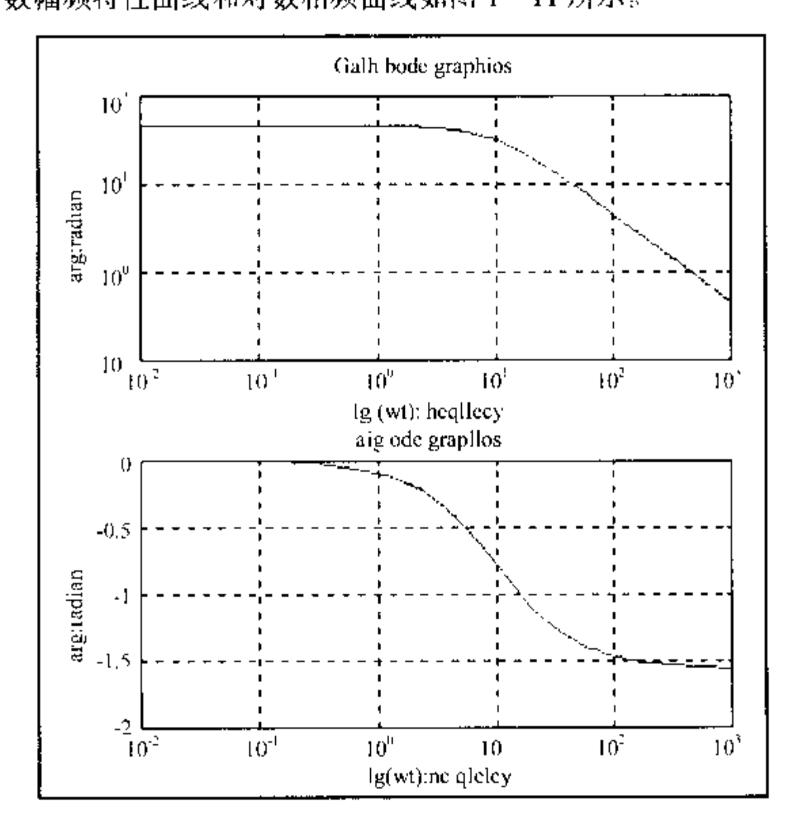


图 1－11 一阶系统对数频率特性图

分析：对数幅频特性曲线是双对数坐标图，而对数相频曲线是 x 轴对数坐标图，可以用 subplot()函数将其绘制在一张图的两个坐标系中，x 轴坐标对齐，便于分析和讨论系统的性能。

求解过程：

在 MATLAB Command Window 中键入下述语句：

```
%关闭所有图形窗并清屏
close all;
clf;
%数据初始化
wt1 = 0.01:0.05:10;
wt2 = 10:10:1000;
%两个向量和为一个
wt = [wt1 wt2];
G = 45./sqrt(1 + 0.01 * wt. * wt);
theta = atan( - 0.1 * wt/1);
%绘图部分
subplot(211),
%双对数坐标系，对数幅频特性曲线
loglog(wt,G);
title('Gain bode graphics');
xlabel('lg(wt):frequency');
ylabel('lg(G(wt)):gain');
grid;
subplot(212),
%x 轴对数坐标系，对数相频曲线
semilogx(wt,theta);
title('arg bode graphics');
xlabel('lg(wt):frequency');
ylabel('arg:radian');
grid;
```

小结：图 1-11 中上边是对数幅频特性曲线，下边是对数相频曲线。因为两图采用同样的横坐标刻度，可以综合分析并评价此一阶系统的性能。从对数幅频特性曲线上可以看出，曲线的拐点在大约 $\omega = 10$ 附近，对应到对数相频曲线上相角弧度值约为 -0.8，化为角度值约为 $-45°$。此拐点即系统的截止频率，约 1.59Hz，说明系统的低通特性比较好，同时带来的问题是响应速度比较慢。不过将幅频特性曲线延伸，系统增益为 -20dB 时对应的相角裕量为 90°，相角裕量很大，有很富裕的空间来增加控制手段以改善系统的稳态增益和快速性。截止频率之前系统的增益约为 1.65dB，截止频率之后系统增益下降的斜率为 -20dB 每十倍频程，即 ω 每扩大十倍系统的增益就下降 20dB。

从对数相频特性图上还可以看出，ω 趋于无穷时相角 arg 趋于 $-90°$，说明此一阶系统是恒稳的。熟悉控制领域的读者可以发现，以上这些通过对数频率曲线得出的结论与直接时域仿真的结果是吻合的。在对数频率曲线的绘制过程中，还有一点需要说明的是：数

据初始化时在截止频率之前和之后选用了不同的数据采样间隔，之前是从 0.01 到 10 每隔 0.05 采一个点，之后是从 10 到 1000 每隔 10 采一个点。这是因为考虑到此一阶系统的特点，即截止频率在低频段，变化相对剧烈。为了精确刻划系统的变化规律，应该在低频段多采样。当然，也可以对原始数据进行对数线性化处理，不过总的说来要麻烦一些。对截止频率的估计不必非常准确，只需在低频段、中频段、高频段大致估计出一个范围即可。

有关 MATLAB 二维图形的绘制就介绍到此，还有一些例如坐标轴修改、颜色、字体设置等功能就不再一一说明了，不知道这些功能并不妨碍读者绘制出完整、美观的二维图形。掌握了上面这些内容，在控制系统设计和仿真领域基本上就可以绘制出令人满意的二维图形，需要了解详细内容的读者可以参阅有关手册或 MATLAB 帮助，本书的附录也有一些相关资料可供查询。

1.4.3 三维图形绘制

与二维图形绘制相比，三维图形绘制在控制系统的应用并不是很多，主要集中在运动轨迹的绘制和信号处理等方面。不过三维空间可以看作是高维空间的示意，借助它可以模拟状态空间的一般情形，对公式进行直观的演示。自从 4.0 版本以后，MATLAB 大大强化了三维图形的绘制功能，有十几条语句可以绘制不同类型的三维曲线、三维网格图、三维曲面图和等高线。熟悉 MATLAB 二维图形绘制的读者可以直接调用与 plot()语句相似的 plot3()语句绘制三维曲线，格式完全相同，只不过多一个数据参数。并且 MATLAB 使漂亮的三维曲面不再只是高级程序员的专利，非常复杂的表达式只消短短几行就可以实现了。表 1-4 是一些常用的三维图形绘制语句：

表 1-4 MATLAB 常用三维图形绘制函数

语 句	功 能	语 句	功 能
plot3()	三维曲线绘制	meshz()	三维网格加围裙
meshgrid()	生成二维平面网格	meshc()	三维网格加等高线
mesh()	三维网格图形绘制	surf()	三维曲面图形绘制
surfc()	三维曲面加等高线	contour()	等高线在 xy 平面投影
waterfall()	x 或 y 方向的水流效果	contour3()	三维空间等高线

应用这些语句，能够方便的绘制出需要的三维曲线、图形。下面通过一个简单的例子，具体介绍上述一些语句的用法。

[例 1.7] 应用 MATLAB 三维曲线和图形绘制语句，绘制下列函数的图形：

(1) $x = t\sin(t), y = t\cos(t), z = t, t \in (-\infty, \infty)$。

(2) $z = 3(1-x)^2 e^{-x^2-(y+1)^2} - 10\left(\frac{x}{5} - x^3 - y^5\right) e^{-x^2-y^2} - \frac{1}{3} e^{-(x+1)^2-y^2}$

结果：第一问曲线见图 1-12；第二问曲线见图 1-13。

求解过程：

在 MATLAB Command Window 中键入下述语句：

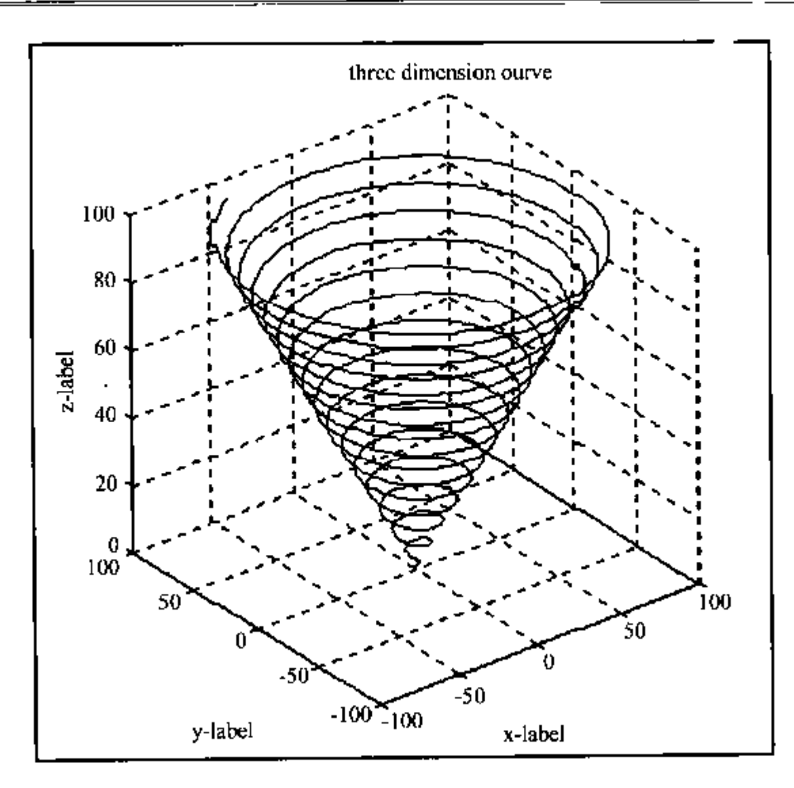


图 1－12 plot3()语句绘制的三维曲线

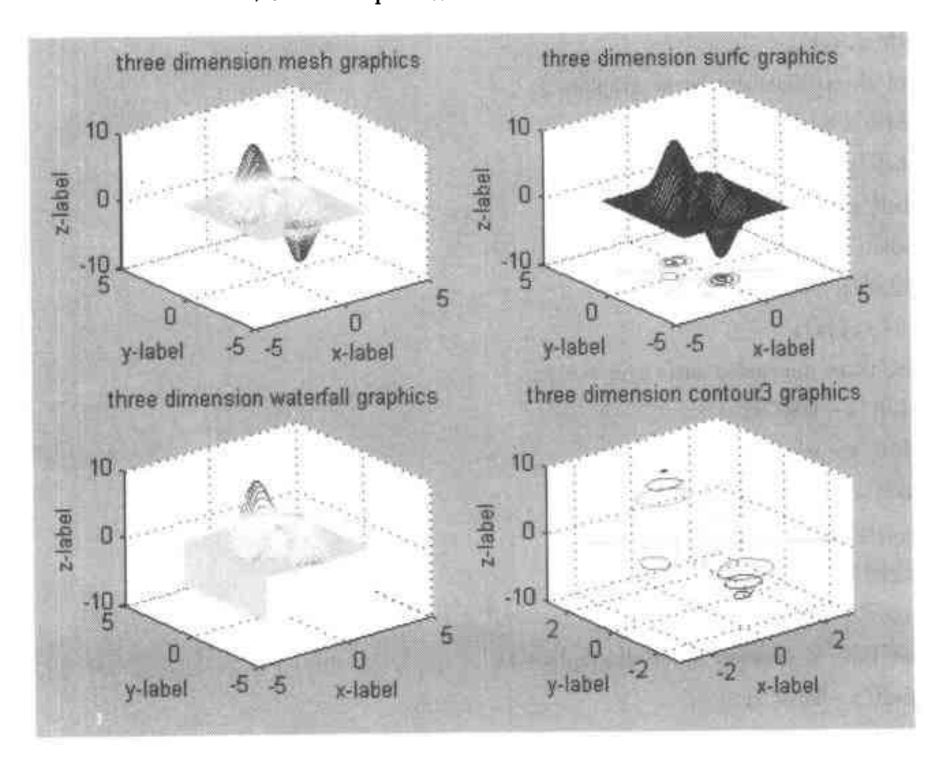


图 1－13 mesh()surfc()waterfall()contour3()绘制的三维图形

第一问,代码如下:

```
%数据初始化
t=0:0.05:100;
```

```
x = t. * sin(t);
y = t. * cos(t);
z = t;
%绘制普通三维曲面
plot3(x,y,z);
title('three dimension curve');
xlabel('x - label');
ylabel('y - label');
zlabel('z - label');
grid;
```

第二问,代码如下:

```
xx = -3:0.1:3;
yy = -3:0.1:3;
%生成三维曲面原始数据
[x,y] = meshgrid(xx,yy);
z = 3 * (1 - x).^2. * exp( - (x.^2) - (y + 1).^2) ...
 - 10 * (x/5 - x.^3 - y.^5). * exp( - x.^2 - y.^2) ...
 - 1/3 * exp( - (x + 1).^2 - y.^2);
subplot(221),
%绘制加网格的三维曲面
mesh(x,y,z);
title('three dimension mesh graphics');
xlabel('x - label');
ylabel('y - label');
zlabel('z - label');
subplot(222),
%绘制加等高线的三维曲面
surfc(x,y,z);
title('three dimension surfc graphics');
xlabel('x - label');
ylabel('y - label');
zlabel('z - label');
subplot(223),
%绘制加流水效果的三维曲面
waterfall(x,y,z);
title('three dimension waterfall graphics');
xlabel('x - label');
ylabel('y - label');
zlabel('z - label');
subplot(224),
%绘制加空间等高线的三维曲线
contour3(x,y,z);
title('three dimension contour3 graphics');
```

```
xlabel('x - label');
ylabel('y - label');
zlabel('z - label');
```

小结：使用本例提到的这些三维图形绘制语句，基本上就可以完成绝大部分三维曲线和图形的绘制工作。前文已经提到，plot3()语句与plot()语句格式相同，使用mesh()语句前先要用meshgrid()语句将x轴和y轴的数据编织成网格。mesh()语句也可以设置颜色参数c，缺省为c=z，也就是颜色与图形高度成正比。contour3()语句绘制的是该图形的三维等高线。通过绘制出来的三维图形可以看出，第二问的函数有三个极大值点，一个极小值点，x和y趋于无穷时z趋于零。事实上，第二问采用的函数解析式是MATLAB提供的，在MATLAB Command Window下直接键入"peaks"就可看到这个函数的MATLAB表达式和用surf()语句绘制的图像。

关于MATLAB二维和三维绘图介绍到这里就告一段落了，这虽然仅仅是一个粗浅的入门，但基本上已经可以胜任控制领域的绘图工作了。当然，还需要读者自己不断地学习和探索。下一节将开始介绍有关MATLAB语言和程序编制的知识。

1.5 MATLAB编程

从严格意义上说，MATLAB并不是一种高级计算机语言，因为它并不生成可执行文件，不能脱离MATLAB环境执行。但从使用效果来看，只要具备了MATLAB环境，使用MATLAB语言编程可以非常简单地实现高级语言具有的某些功能，尤其是关于科学计算的部分，甚至比绝大多数高级语言实现还要简单。因此，在这种意义下，MATLAB不但是一种交互计算环境，还是一种功能强大的程序语言。

目前流行的MATLAB 5.x版本还提供了MATLAB Compiler，此编译器可以将输入的M文件转化为C源代码，并且，有关C/C++的接口和数学库的功能也在逐步完善。MATLAB语言越来越独立和丰富，或许有一天，MATLAB终究会发展成一种标准，一种科学家和工程师通用的计算机高级语言。

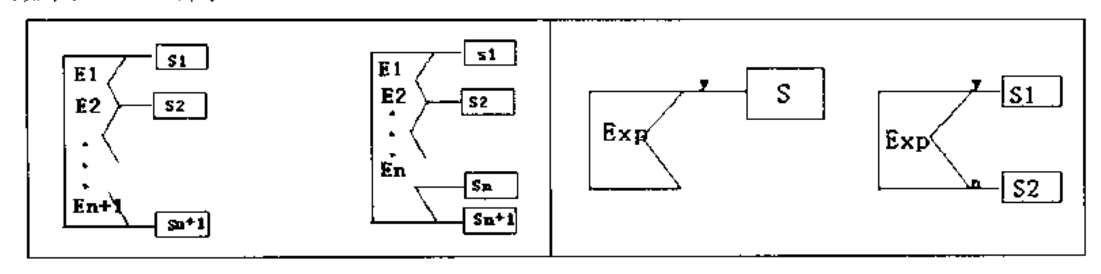
MATLAB的控制语句与C语言的格式非常类似，常用的if-else、while、switch-case、try-catch、return等语句与C语言中对应语句的用法基本一致。甚至fprintf()、fopen()等I/O函数的调用格式也与C语言相同。MATLAB 5.x版本还引入了结构的概念，熟悉C语言中结构概念的读者也是不难理解的。事实上，有了任何一门高级语言基础都可以非常轻松地掌握MATLAB语言的控制语句和编程技巧。以前没有接触过计算机语言也没有关系，通过本节的讲述和示例能够很快地熟悉这些概念。

一方面是为了保持内容的完整性，另一方面让不熟悉的读者有一个了解的过程，本节将简要介绍MATLAB的控制语句以及M文件、M函数的编写和运行。这也是使用MATLAB开发控制系统应用软件的基础

1.5.1 条件转移语句与逻辑表达式

本书前面所述的M文件和语句都是顺序结构的，条件转移语句是改变顺序结构的基本方式之一。条件转移语句一般可以分为二分支结构和多分支结构，其语句结构的PAD

图如图 1－14 所示。



(a)二分支结构　　(b)多分支结构

图 1－14 条件转移语句 PAD 图

图中"Exp"代表逻辑表达式，"S"代表语句。二分支结构的含义是：程序执行到此，首先对逻辑表达式进行判断，如果逻辑值为真，则执行语句段"S1"；反之则执行语句段"S2"，以此来实现对程序结构的控制。多分支结构的含义是：程序执行到此将主语句的表达式（一般不是逻辑表达式）与各分语句的表达式进行匹配，完全匹配后则执行相应分语句段；如果不存在完全匹配则执行缺省的分语句段，或是跳过此多分支结构执行下边的语句。当然，多分支结构可以由多个二分支结构嵌套来实现，但这样一来流程图就变得相当复杂，可读性大大降低。

对应于这两种条件语句结构，和 C 语言一样，MATLAB 也给出了 if－else 和 switch－case 两条语句来完成上述功能。其中 if－else 结构的标准形式为：

```
IF expression
    statements
ELSEIF expression
    statements
ELSE
    statements
END
```

switch－case 结构的标准形式为：

```
SWITCH switch_expr
CASE case_expr,
    statement, ..., statement
CASE {case_expr1, case_expr2, case_expr3,...}
    statement, ..., statement
...
OTHERWISE,
    statement, ..., statement
END
```

和 C 语言稍有不同的是，这两种语句结构都使用了"END"语句作为结构体的结尾，这是因为 MATLAB 中没有表示结构体的"{}"。当然，熟悉 Visual Basic 的读者可能对这种格式更习惯一些。

这两种语句结构中的"expression"指的是逻辑表达式，即用关系操作符连接起来的表达式。在 MATLAB Command Window 下键入"relop"（Relational Operators）可以看到

MATLAB 认可下述 10 种关系操作符：

表 1-5 MATLAB 关系操作符

关系操作符名	功能	关系操作符名	功能	关系操作符名	功能
>	大于	= =	等于	|	逻辑或
> =	大于等于	~ =	不等于	~	逻辑非
<	小于	&	逻辑与	xor	逻辑异或
< =	小于等于				

需要注意的是判断等于的关系操作符和 C 语言一样，是两个等号“= =”。如果使用一个等号“=”，那么 C 编译器就会当作是赋值语句而将其逻辑值永远判断为真，从而有可能得出错误的结果。而 MATLAB 编译器则会给出错误提示：“A closing right parenthesis is missing. Check for a missing ")" or a missing operator.”。

在 switch - case 语句结构中，与 C 语言不同的是每个 case 语句后面不必跟随一个 break 语句，MATLAB 自动寻找下一个 case 语句。并且，缺省语句段的标识也不是“default”，而是“otherwise”。

有关上述两种条件转移语句的功能相信有高级语言基础的读者都已经比较熟悉，这里就不再赘述了。下面以一个简单的例子介绍一下它们在 MATLAB 中的用法。

随机产生一个 0 到 100 之间的整数，判断能否被 16、4、2 整除。分别用 if - else 结构和 switch - case 结构实现。

求解过程：

第一问，if - else 结构实现：

在 MATLAB Command Window 中键入下述语句：

```
%清除内存变量
clear;
%数据初始化
%产生(0,1)间随机数并乘以 100 取整
x = ceil(rand(1) * 100);
%if - elseif - else - end 条件转移语句结构
%判断能否被 16 整除
if rem(x,16) = = 0
    disp('it can be divided by 16');
elseif rem(x,4) = = 0
    disp('it can be divided by 4');
elseif rem(x,2) = = 0
    disp('it can be divided by 2');
else
    disp('it cannot be divided by 16,4 and 2');
end
x
```

结果为:

```
it cannot be divided by 16,4 and 2
x = 77
```

第二问,switch - case 结构实现:

在 MATLAB Command Window 中键入下述语句:

```
%消除内存变量
clear;
%数据初始化
x = ceil(rand(1) * 100);
%switch - case - otherwise - end 条件转移语句结构
%判断除以 16 后的余数
switch rem(x,16)
case 0 ,
      disp('it can be divided by 16')
case {8,4},
      disp('it can be divided by 4')
case {14,12,10,6,2},
      disp('it can be divided by 2')
otherwise
      disp('it cannot be divided by 16,4 and 2')
end
x
```

结果为:

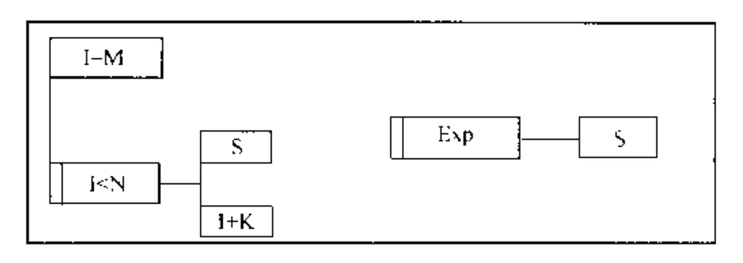
```
it can be divided by 2
x = 46
```

小结:本题使用了"clear"语句,其功能是清除所有内存变量,以防第一问的结果对第二问产生影响。关于语句本身有两点需要说明:其一是逻辑表达式不需要加括号,括号只是表示运算的有限级;其二是在 switch - case 结构中,如果希望若干个结果都执行相同的 case 语句,可以用"{}"将这些结果括起来,放在一个 case 语句后面。比较这两种不同的实现方式,明显可以发现从程序流程的可读性上来说,switch - case 语句结构要比 if - else 语句结构更加清晰。也就是说,虽然 if - else 这种二分支语句结构可以实现 switch - case 多分支结构的功能,但并不提倡嵌套使用 if - else 结构。

1.5.2 循环语句

循环语句是高级语言控制语句的一个最基本的概念,MATLAB 解释的含义是重复执行某一段表达式若干次,该次数可以是指定的,也可以是不定的。这就给出了两种格式的循环语句,如图 1 - 15 所示。

在指定次数的循环中,有一个循环变量统计循环的次数,对应于图 1 - 5(a)中的 I。循环表达式由三部分组成,即循环初值、循环步长、循环终值,分别对应于上图的 M、K、N,这些循环值可以是数值也可以是表达式。程序进入循环后执行顺序是:首先将循环初值赋给循环变量,然后和循环终值比较,小于循环终值则执行循环体,否则跳出循环体。



(a)指定次数循环 PAD 图　(b)不定次数循环 PAD 图

图 1－15　循环语句 PAD 结构图

执行完循环体后将循环变量加上循环步长后再赋给循环变量,并开始下一次循环

在不定次数循环中,其循环表达式一般为逻辑表达式。程序进入循环后执行顺序为:首先判断循环表达式的逻辑值,若为真则执行循环体,否则跳出循环体　执行完循环体后再判断循环表达式的逻辑值,进入下一次循环。

与 C 语言一样,MATLAB 也提供了两条语句分别实现上述两种循环的功能,即 for 语句和 while 语句。其中 for 语句的标准格式为:

```
FOR variable = expr,
    statement,
    ...,
    statement
END
```

MATLAB 特别强调了用逗号“,”隔开各循环体语句可以提高使用内存的效率。

while 语句的标准格式为:

```
WHILE expression
    statements
END
```

这里的“statements”表示循环体中各语句之间用分号隔开,“expression”也是逻辑表达式,表 1－5 的关系操作符在这里同样适用。

同样需要注意的是表示循环体结束时用的是“END”语句,各表达式也不用括号括起来。for 语句和 while 语句还有一条辅助语句,即 break 语句。其功能是中止当前的循环体,执行本循环体之外的语句。也就是说,如果在循环嵌套的最里层使用“break”语句的话,只能跳出本层的循环体,继续执行外层的循环。

关于这两条循环语句的功能就不再赘述了,下面以一个简单的例子使读者熟悉一下它们在 MATLAB 中的用法。

试用 Taylor 展开法求解下述矩阵的指数,并与 MATLAB 提供的函数 expm( )求解结果进行比较。

$$A=\begin{pmatrix}1&2&3&4\\5&6&7&8\\1&3&5&7\\2&4&6&8\end{pmatrix}$$

分析:Taylor 展开法即矩阵指数的原始定义,见下式。

$$e^{At} = I_n + At + \frac{1}{2!}A^2t^2 + \cdots + \frac{1}{k!}A^kt^k + \cdots = \sum_{k=0}^{\infty}\frac{A^kt^k}{k!}$$

用它来求解的优点是便于控制展开的阶次和精度.下面分别用 for 语句和 while 语句实现。

求解过程:

第一问,for 循环结构实现:

在 MATLAB Command Window 中键入下述语句:

```
%清除内存变量
clear;
%数据初始化
A=[1,2,3,4;5,6,7,8;1,3,5,7;2,4,6,8];
E=zeros(size(A));
S=eye(size(A));
%for 循环结构
for k=1:100
E=E+S,
S=A*S/k,
if norm(S,1)<0.00001 break;
end
E
```

结果为:

```
E  =  1.0e+007  *
      0.9929      1.7481      2.5034      3.2586
      2.4927      4.3888      6.2848      8.1809
      1.6108      2.8361      4.0614      5.2866
      1.9858      3.4963      5.0067      6.5172
```

第二问,while 循环结构实现:

在 MATLAB Command Window 中键入下述语句:

```
%清除内存变量
clear;
%数据初始化
A=[1,2,3,4;5,6,7,8;1,3,5,7;2,4,6,8];
E=zeros(size(A));
S=eye(size(A));
k=1;
% while 循环结构
while norm(S,1)>0.00001
E=E+S,
S=A*S/k,
k=k+1
end
E
```

结果为:

```
E = 1.0e+007 *
      0.9929    1.7481    2.5034    3.2586
      2.4927    4.3888    6.2848    8.1809
      1.6108    2.8361    4.0614    5.2866
      1.9858    3.4963    5.0067    6.5172
```

MATLAB 提供的 expm()函数:

```
expm(A)
ans = 1.0e+007 *
        0.9929      1.7481      2.5034      3.2586
        2.4927      4.3888      6.2848      8.1809
        1.6108      2.8361      4.0614      5.2866
        1.9858      3.4963      5.0067      6.5172
```

小结:从结果可以看出,使用 Taylor 展开法求得的矩阵指数与 MATLAB 提供的 expm()函数求得的值是完全一致的。使用 for 循环时,将展开式的阶次限定在 100 次,对于数值计算来说,这已经是很高的精度了。如果精度达到了 0.00001,则用"break"语句跳出循环体。使用 while 循环时,精度定为 0.00001,这是因为如果不作特殊规定,MATLAB 将保留结果的小数点后四位。

1.5.3 M 文件及 M 函数的编写与运行

在本节里,我们将正式接触到 MATLAB 环境下编程的整个过程,包括 M 文件与 M 函数、MATLAB 变量管理、系统调用等一系列基本知识。通过本节的阅读,读者能够对 MATLAB 语言环境有一个整体了解,更加充分地发挥 MATLAB 语言的潜能,编制出更标准的程序。

事实上,本书前述的所有例程都可以称作 M 文件。在 MATLAB Editor/Debugger 下键入这些语句并存盘,就可以得到一个 .m 后缀的文本文件,在 MATLAB Command Window 下直接键入该文件名即可执行该文件。和 M 文件不同,M 函数不能通过直接键入其文件名来执行,必须通过提供参数的调用格式才能使用。前边介绍的绝大部分函数都是 M 函数,如求取矩阵条件数的 cond()函数,绘制二维图形的 plot()函数等等。MATLAB 中还有一类函数称为内部函数,这些函数是无法读取的,但作为使用者来说,它与 M 函数并没有什么区别。例如前文提到的 expm()函数和 expm1()函数,它们完成相同的功能,但 expm()是内部函数,而 expm1()是 M 函数。

在以后的叙述中,我们就将 M 文件的编写和调试从 MATLAB Command Window 转移到 MATLAB Editor/Debugger 下进行(图 1-16)。在这个集成环境中可以方便地进行新建、修改、存储,选择 Tools 菜单中的 Run 命令就可运行程序,结果显示在 MATLAB Command Window 中。

选择 Debug 菜单就可以进行调试,高级语言常用的 Set/Clear Breakpoint、Single Step、Stop if error 等选项这里也都有,可以很方便地调试程序。

M 函数与 C 语言中的库函数有些类似,读者通过自己编写 M 函数可以丰富 MATLAB 的功能,满足特殊的需要。事实上,MATLAB 功能繁多的工具箱就是一个个 M 函数

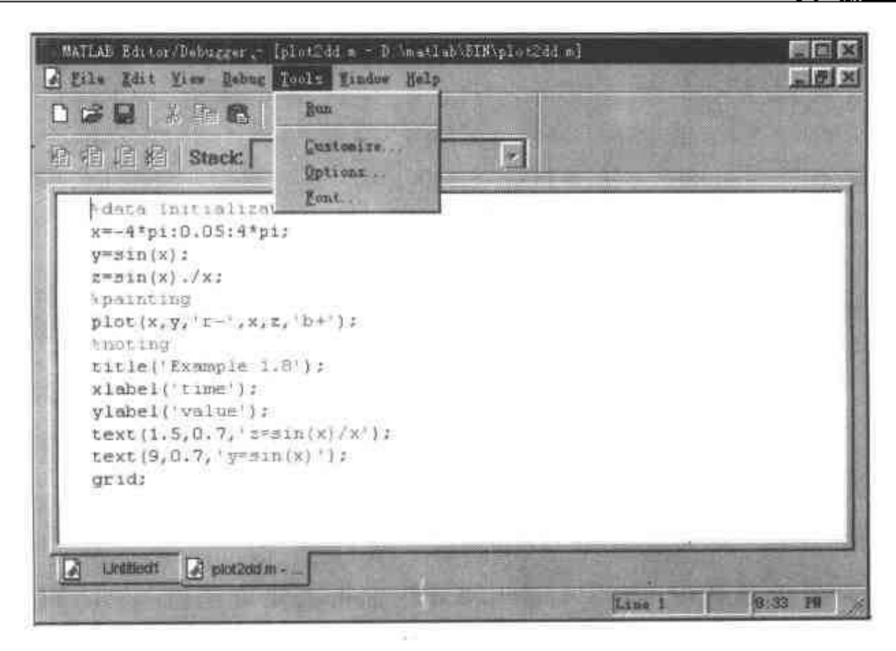


图 1-16 M 文件的运行:MATLAB Editor/Debugger

的集合,读者也可以通过这种方法编写自己的工具箱。M 函数的第一行必须是以"function"语句开头,紧接着是返回值列表、函数名、参数列表。与 C 语言的函数不同,M 函数的返回值可以一个也可以是多个,而且不局限于数值,归根结底,它们都是矩阵。一般说来,M 函数从第二行开始是函数注释,这并不是传统的随行注释,而是整块的对函数的解释。并且,注释应写得尽量详细,因为在 MATLAB Command Window 下键入 help 加该函数名,就可以看到此注释的内容。例如我们在 MATLAB Editor/Debugger 下打开名为 expm2.m 的 M 函数,即可看到如下的代码:

```
function E = expm2(A)
%EXPM2 Matrix exponential via Taylor series.
% E = expm2(A) illustrates the classic definition for the
% matrix exponential. As a practical numerical method,
% this is often slow and inaccurate.
%
% See also EXPM, EXPM1, EXPM3.

% Copyright (c) 1984-98 by The MathWorks, Inc.
% $Revision: 5.4 $ $Date: 1997/11/21 23:38:26 $

% Taylor series for exp(A)
E = zeros(size(A));
F = eye(size(A));
k = 1;
while norm(E+F-E,1) > 0
E = E + F;
F = A*F/k;
```

```
k = k + 1;
end
```

在 MATLAB Command Window 下键入 help expm2，即可看到：

```
help expm2
%帮助信息
EXPM2 Matrix exponential via Taylor series.
E = expm2(A) illustrates the classic definition for the
    matrix exponential. As a practical numerical method,
    this is often slow and inaccurate.
See also EXPM, EXPM1, EXPM3.
```

细心的读者肯定会发现，键入 help expm2 后显示的帮助信息只是该 M 函数注释的一部分，这是因为原注释中间有一个空行。如果你不希望使用者看到某些注释信息，例如作者、版本号等，也可以采用这种办法。有关 M 函数的调用想必大家已经很熟悉了，这里就不再赘述了。

用户进入 MATLAB 环境后，MATLAB 就为其建立一个工作空间，所有的变量信息都存储在此工作空间中。用户可以使用"who"和"whos"命令随时查看。例如运行完前例 1.12 的程序后，键入"who"命令，有：

```
who
Your variables are:
x xx y yy z
```

键入"whos"命令可以查看这些变量的具体信息：

```
whos
%变量信息
Name      Size      Bytes        Class
x         61x61     29768        double array
xx         1x61       488        double array
y         61x61     29768        double array
yy         1x61       488        double array
z         61x61     29768        double array
Grand total is 11285 elements using 90280 bytes
```

这些信息包括变量名、维数、占用空间、变量类型以及总的元素数和占用空间。通过上边的信息可知，本工作空间共有 11 285 个元素，占用内存为 90 280 个字节。

想要清除某个内存变量，可以使用"clear"命令，紧接上例，如：

```
clear x
who
Your variables are:
xx y yy z
whos
%变量信息
Name      Size      Bytes        Class
xx         1x61       488        double array
y         61x61     29 768       double array
```

```
yy        1x61       488       double array
z         61x61      29 768    double array0
Grand total is 7564 elements using 60512 bytes
```

如果直接使用"clear"命令不带参数,则清除所有内存变量,这一功能在前边的叙述中已经提到了。其目的是初始化工作空间,重新开始运算。

在 MATLAB 中可以直接调用可执行文件或 DOS 命令。其方法是在 MATLAB Command Window 下键入!加可执行文件名。如果不是系统命令则要保证该系统文件在缺省目录下,如果是 DOS 命令就不必有此要求。如:

```
! dir *.m
%驱动器信息
Volume in drive D is SW
Volume Serial Number is 2A55 - 13D4
Directory of D: \ matlab \ BIN
%文件索引
ADJACENT M       11 523       12 - 05 - 98      17:59 adjacent.m
FORWARD M        9 866        12 - 28 - 98      18:18 forward.m
MECHSYS M        85           07 - 25 - 99      17:11 mechsys.m
3file(s)         21 474 bytes
0 dir(s)         176 455 680 bytes free
```

通过 DOS 命令 dir 我们就可以看到 Directory of D: \ matlab \ BIN 目录下的文件信息了,其它 DOS 命令在 MATLAB 下的用法也与此类似,这里就不再赘述了。

1.6 小　结

本章的主要内容就到此为止,我们主要介绍了 MATLAB 在矩阵运算、绘图、程序编制等方面的优点和使用方法,为介入 MATLAB 在控制系统设计和仿真领域的应用打下一个基础。对于以前就熟悉 MATLAB 的读者,本章的内容可以作为复习和参考资料随时查阅;而对于以前没有接触过 MATLAB 的读者,通过仔细阅读本章的内容可以大致了解 MATLAB 数值计算的风格,熟悉 MATLAB 环境。

作为控制系统设计和仿真领域的基础知识,本章的关键内容是矩阵运算和图形绘制。矩阵运算是控制领域学习和实践的基础,图形绘制则是应用 MATLAB 进行控制系统设计和仿真的重要手段。从下一章开始,我们将进入正题,具体讲述有关 MATLAB 环境下控制系统设计和仿真方面的内容。

第二章 控制系统的数学描述

从本章开始,我们正式进入到有关 MATLAB 在控制系统设计和仿真领域的应用中去。首先来回顾一下控制系统的概念及其发展。本世纪初以来,特别是从第二次世界大战以来,控制科学和控制技术得到了迅速发展。自动控制极大地提高了劳动生产率和产品质量,推动了现代工业的巨大进步。在军事上,控制技术有效地提高了武器的精确度和威力。在航天、制导、核能等方面,控制技术更是不可缺少的。在工业和军事领域中,控制技术的作用是:不需要人的直接参与,而控制某些物理量按照指定的规律变化。

控制理论研究问题是:(1)一个给定的控制系统,它的运动有哪些性质和特征?(2)怎样设计一个控制系统,使它们运动具有给定的性质和特征?前一个问题称为分析,后一个问题称为综合和设计。对于这两个控制理论研究的基本问题,我们将在以后的各章节中详细论述。本章要讨论的问题是,怎样找出适合控制系统数学描述形式。

众所周知,运动是宇宙存在的普遍形式。我们对控制系统的研究,就离不开对控制系统的运动的研究。所谓运动,并不只是指未知的移动和旋转,而是泛指一切物理量随时间的变化,如温度的升降、电流的强弱、人口的增减等。要研究各种物理量的变化,必须把它们彼此之间相互作用的关系和各自的变化规律用数学形式描述出来。这就是常说的为某一事物建立数学模型。建立描述控制系统运动的数学模型,是控制理论的基础。一般控制系统数学模型的建立都是基于图 2-1 的思想。

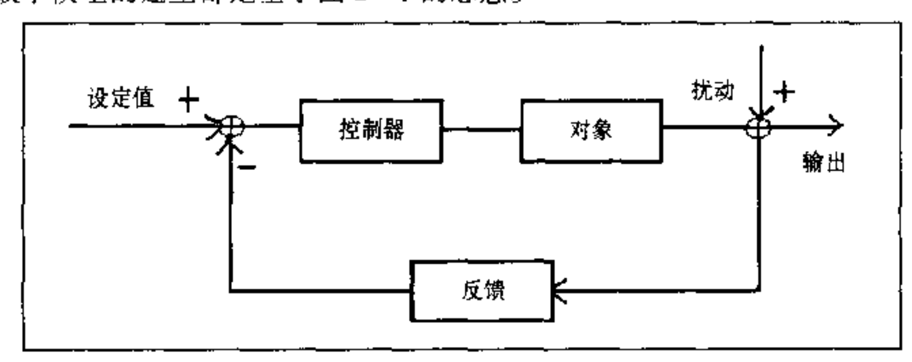


图 2-1 控制系统的模型

图 2-1 是基于偏差控制的反馈控制系统简图。设定值与反馈量的偏差通过控制器作用于被控对象,经过扰动的作用,最终达到控制输出量跟随设定值的目的。对于本图来说,建立数学模型就是要找到一种合适的手段,详细描述被控对象的运动规律,从而为控制器和反馈系数的设计提供可靠的依据。

描述控制系统的数学模型的形式不只一种。它们各有特长和最适用的场合。它们彼此之间也有紧密的联系。大致说来,主要分为两种:一种称为控制系统的时间域描述;另一种称为控制系统的频率域描述。所谓时间域描述,是指用微分方程和状态方程来描述

控制系统,从时间推演的角度刻划系统的运动。而频率域描述,是指用 Laplace(拉普拉斯)变换工具来描述控制系统,即通过传递函数从频率变化的角度刻划系统的运动。事实上,这两种数学描述方法的共同基础都是微分方程。状态方程就是一组联立的一阶微分方程,拉普拉斯变换也是在系统微分方程的基础上进行的。因此,本章从系统微分方程的列写和求解入手,详细讨论 MATLAB 在建立系统数学模型方面的应用,并给出大量的实例和一些较为成熟的算法,使读者能够迅速掌握其要领。

2.1 控制系统的运动方程

从工程的角度来说,描写控制系统运动的最基本的数学工具就是微分方程。

要写出描述控制系统运动的微分方程,大致分为两种基本方法。第一种方法是分析系统各部分运动的机理,根据这些机理分别写出描述各部分运动的微分方程,合在一起便成为描述整个系统的方程。第二种方法是人为的在系统上加上某种测试信号,记录系统中各变量的运动,然后选择适当的微分方程,使之能近似地表示这种运动,以此作为系统的方程。有时连测试的信号也不加,就径直记录系统运行时各变量的实际运动,据以建立数学模型。这种方法称为系统辨识方法,主要用于系统的运动机理复杂因而不便分析和不可能分析的情况。

本章的讨论以第一种方法为主,适当兼顾第二种方法,并且主要通过实例来说明。首先我们从线性对象入手,即可以用线性微分方程和线性代数方程描述的对象,并且限定于定常的和集总参数的对象,即其参数不随时间变化,各物理量不随空间位置变化的对象。这种线性、定常、集总参数的对象是工程上最常见的。

2.1.1 微分方程数值解

有关系统微分方程列写的问题,读者可以在任何一本讲述控制理论的教材中找到,其数学基础在工科大学的数学课程中都有所介绍。事实上,除了数学上的熟练程度之外,这还涉及到对所接触的实际问题的理解和系统分析的能力。这并不是本书所能解决的问题,也不是 MATLAB 的所长。因此,除了少数特殊说明的情况之外,本书均假设系统的运动方程或传递函数已经得到。

熟悉信号与系统分析的读者都清楚,得到系统的微分方程之后,通过拉普拉斯变换和反变换,就可以得到线性时不变方程的解析解。对于状态方程,也可以用状态转移矩阵 $\Phi(t)$ 求解。解析解是精确的,然而通常在计算上存在困难,或根本是不可能的。因此,人们寻找了许多关于微分方程的数值解法,力求在条件允许的范围内逼近方程的解析解。MATLAB 在寻找系统微分方程的数值解方面作的非常出色,MATLAB5.2 提供了三个采用龙格-库塔法(Runge-Kutta)求解微分方程数值解的函数,分别是 ode23()、ode45()、ode113()(4.x 的版本没有 ode113()函数),对应不同的精度。其中 ode113()函数精度最高,其基本调用格式为:

```
[T,Y] = ODE113('F',TSPAN,Y0,OPTIONS)
```

其中 T 是时间向量,Y 是与 T 相互对应的方程的数值解;'F'是对于系统微分方程的描述,一般包含在特定的 ODEfile 中;TSPAN 是 1×2 的向量[T0 TFINAL],表示仿真的

开始和结束的时间；Y0 是该微分方程的起始条件；OPTIONS 是控制精度的可选参数，由 odeset()来设置，常用的参数值有'RelTol'，表示 1e－3 精度，或'AbsTol'，表示 1e－6 精度。

单有 ode113()函数是无法求解微分方程的，必须配合 odefile()函数使用。odefile()函数的功能是描述系统的微分方程和一些相关参数，提供 ode113()函数的接口。其基本的调用格式为：

```
function F = odefile(t,y)

F = < Insert a function of t and/or y here. >;
```

有关系统微分方程的描述，需要说明的是：ode113()函数只接受一阶微分方程的形式。因此，对于高阶微分方程，首先要化为若干个一阶微分方程，然后再使用 odefile()函数。事实上，odefile()函数能够完成的功能很多，例如起始条件的设置、求解该微分方程的雅可比矩阵(Jacobian Matrix)、数值解过零区域的设置等等，都可以提供调用不同格式的 odefile()函数来完成。ode113()函数也有许多衍生的函数。有关这方面的情况就不再一一赘述了，有特殊需要的读者请参阅 MATLAB 帮助。下面以一个简单的例子，使读者熟悉一下 ode113()函数和 odefile()函数的基本用法，掌握使用 MATLAB 求解系统微分方程数值解的基本步骤。

［例 2.1］　某机械运动系统如图 2－2 所示，物体质量 M＝1kg，弹簧的虎克系数为常数 K＝20N/m，摩擦系数也为常数 B＝5N/m/s。t＝0 时，施加外力 f(t)＝30N，试给出系统的运动方程，何时达到稳态，并画出该机械系统位移、速度随时间变化的坐标曲线以及速度与位移的关系曲线。

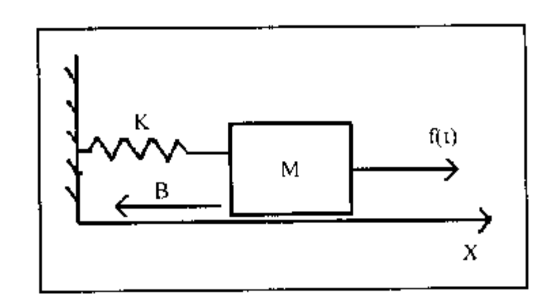


图 2－2　机械系统示意图

结果：根据牛顿运动定理，可以建立系统的运动方程为：

$$M\frac{d^2x}{dt^2}+B\frac{dx}{dt}+Kx=f(t)$$

其中 x 表示物体的位移，其一阶导数和二阶导数分别是物体的速度和加速度。

系统的稳定点为：

```
system values in the stable state:
Time is 2.407091 Displacement is 1.502046 Velocity is 0.009417
```

系统仿真图：

其横坐标为时间，纵坐标为弹簧系统的位移时间响应，如图 2－3 所示。

其横坐标为时间，纵坐标为弹簧系统的速度时间响应，如图 2－4 所示。

其横坐标为弹簧系统的位移，纵坐标为速度，如图 2－5 所示。

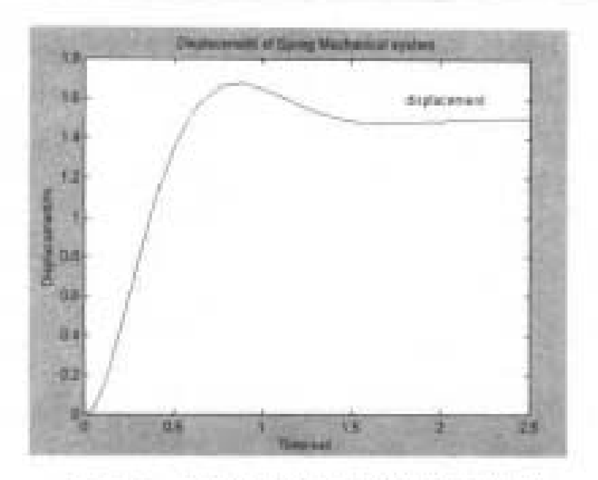


图 2-3 弹簧机械系统位移时间响应曲线

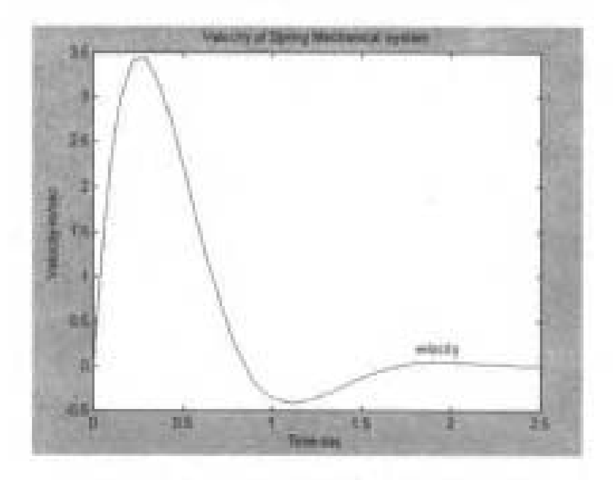


图 2-4 弹簧机械系统速度时间响应曲线

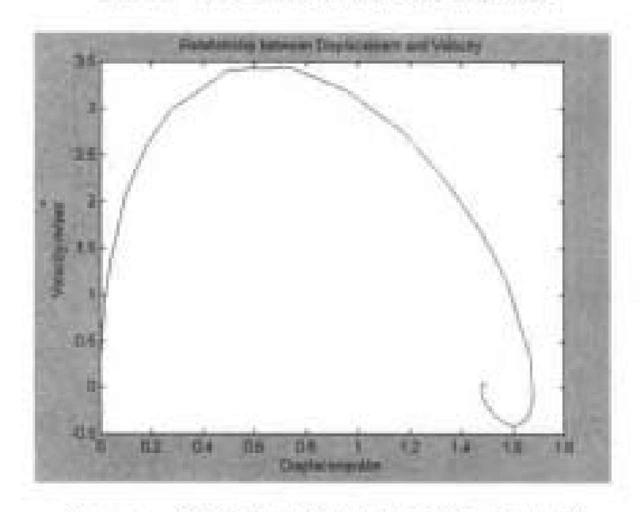


图 2-5 弹簧机械系统速度与位移间关系曲线

求解过程：

为得到上图的结果，在 MATLAB 环境下共分三步进行：

1. 编写系统微分方程

在 MATLAB Command Window 下键入"edit"或选择"File"菜单新建 M－file，进入

MATLAB Editor/Debugger，编辑 M 函数“springsys”：

```
function xt = springsys(t,x)
ft = 30;
M = 1;
B = 5;
K = 20;
xt = [x(2);1/M * (ft - B * x(2) - K * x(1))];
```

选择 MATLAB Editor/Debugger 的“File”菜单的“Save”选项，保存文件名为“springsys.m”。

2. 编写系统的仿真程序

选择“File”菜单新建 M - file，键入以下代码：

```
%关闭所有图形窗口
close all
%数据初始化
ft = 30;
M = 1;
K = 20;
B = 5;
t0 = 0;
tfinal = 2.5;
%仿真开始和结束时间
tspan = [t0 tfinal];
%系统初始条件
x0 = [0,0];
%设置 ode113()函数的可选参数
options = odeset('AbsTol',[1e-6;1e-6]);
[t,x] = ode113('springsys',tspan,x0,options);
%系统的加速度
a = 1/M * (ft - B * x(:,2) - K * x(:,1));
%寻找系统的平衡点
i = 1;
while (abs(a(i)) > 0.1)|(abs(x(i,2)) > 0.01)
    i = i + 1;
end
%显示寻找结果
disp('system values in the stable state:');
result = sprintf('Time is %f \ n',t(i));
disp(result);
result = sprintf('Displacement is %f \ nVelocity is %f',x(i,1),x(i,2));
disp(result);
%绘图
%位移向量
```

```
d = x(:,1);
%速度向量
v = x(:,2);
%绘制时间-位移图
plot(t,d);
title('Displacement of Spring Mechanical system');
xlabel('Time - sec');
ylabel('Displacement/m');
text(1.8,1.6,'displacement');
%绘制时间-速度图
plot(t,v);
title('Velocity of Spring Mechanical system');
xlabel('Time - sec');
ylabel('Velocity - m/sec');
text(1.8,0.2,'velocity');
%绘制位移-速度图
plot(d,v);
title('Relationship between Displacement and Velocity');
xlabel('Displacement/m');
ylabel('Velocity - m/sec');
```

选择“File”菜单的“Save”选项，保存文件名为“spring.m”，注意和 springsys.m 保存在同一路径下。

3.运行

选择“Tools”菜单的“Run”选项或在 MATLAB Command Window 下直接键入文件名 spring，在 MATLAB Command Window 下查看运行的结果。

分析：如前所述，ode113()函数只能求解形如 $y' = F(y,t)$ 格式的一阶微分方程的数值解。因此，需要将本题的二阶微分方程化为两个一阶微分方程的格式，然后再求解。根据系统的运动方程：

$$M\frac{d^2x}{dt^2} + B\frac{dx}{dt} + Kx = f(t)$$

令：

$$x_1 = x, x_2 = \frac{dx}{dt}$$

有：

$$\frac{dx_1}{dt} = x_2$$

$$\frac{dx_2}{dt} = \frac{1}{M}(f(t) - Bx_2 - Kx_1)$$

这样以来，就可以应用 ode113 函数对这两个一阶微分方程进行仿真了。

关于 odefile()函数的使用，需要说明的是与其他函数不同，odefile()其实是一种编写 M 函数的格式，不过这种 M 函数只能配合 ode 函数使用。读者仔细阅读上文的 springsys.m 文件的代码，就可以理解 odefile()函数的使用方法。

通过图 2-5 的系统速度与位移的关系曲线可以看出，系统有一个平衡点，并且是经

过小幅振荡后逐渐趋于平衡点的。从其它仿真结果来看,系统最终的平衡点在距原点1.5m处左右,此时速度和加速度均接近于零。当然,严格说来速度和加速度都不可能达到零,但在工程上进入平衡区域上下限的5%以内就可以认为达到稳态了。因为系统是从零状态出发的,并且有过零的区域,所以判断是否达到稳态时使用了加速度和速度的双重判据,只有在速度和加速度均接近于零时才判定达到稳态。从控制理论的角度来看,如果把稳态的位移看作希望加以控制的输出量,那么物体、虎克系数和摩擦系数就组成被控对象的固有特性,外力 f(t)就是控制手段。通过施加不同的外力就可以期望得到不同的稳态位移。当然,这只是开环控制,抗干扰的能力很差,例如虎克系数或摩擦系数的微小变化就会造成平衡点的较大偏移。如果希望得到较好的系统性能和抗干扰能力,可以把当前位移与期望位移的偏差以电信号或别的形式反馈给施加外力的机构,从而形成本章开始提出的按偏差控制的反馈控制系统模型。

小结:这是我们第一次接触一个完整的控制系统的例子,因此讲述得比较详细。包括M函数及文件的编写和调试、运行、存盘等等。当然,这个例子还是比较简单的,程序也不是很长,结构和解题思路也都比较清晰,读者比较容易接受。除了 ode113()函数之外,ode23()、ode45()、ode15s()、ode23s()、ode23t()、ode23tb()等函数也可以用来进行系统微分方程的仿真,并且能够满足不同的需求。程序中使用了一些以前没有涉及的 MATLAB 函数,例如“sprintf”等等,一方面因为不是解题的重点,另一方面这些函数的含义也比较容易理解,所以其具体使用方法就不在这里赘述了,读者可以自己查阅相关的 MATLAB 帮助。以后的例子也是采取这种方式,当然,如果是比较关键或难于理解的函数,还是会适当解释的。

2.1.2 非线性系统描述

严格说来,所有的控制系统都是非线性的。包括上一节分析的弹簧机械系统,当位移和速度变化时虎克系数和摩擦系数也会产生变化,而且这种变化是位移和速度的函数,这就在系统的微分方程中引入了非线性的因素。但是我们为什么要用线性的模型来分析它呢?这是因为在一定的范围内,这些物理量的非线性变化非常小,按照线性的模型进行控制是精度所允许的,与实际的系统吻合的非常好。并且,系统的线性模型是我们进行理论研究的最重要的手段。这就像物理学的质点和信号处理中白噪声的概念一样,虽然在现实世界中是不存在的,但它是整个理论研究的基石。

还有一些系统,非线性的因素占据主导地位。对于这样的系统,首先要给出其非线性的模型,然后再进行仿真或进行线性化处理,否则是无法揭示系统的本质的。因此,研究非线性系统的运动是非常必要并且十分重要的。描述非线性系统比较传统的方法有描述函数法、相平面法和波波夫法(Popov)等等,这些方法的特点都是以线性模型来近似非线性模型或是直观的描述,作为理论分析是很有用处的。但由于数字式计算机的兴起,除了定性的分析之外,这些方法在工程实际中已经很少使用了。事实上,我们可以对非线性模型直接仿真。MATLAB 提供的 ode 系列函数同样可以进行非线性系统仿真,其调用方式和格式与线性系统完全一样,请看例 2.2:

[例 2.2] 如图 2-6 所示的单摆,绳长 L = 1m,物体质量 M = 1kg,运动阻尼系数 B = 0.1kg/m/s。试给出系统的运动方程,并画出系统位移、角速度随时间变化的坐标图以

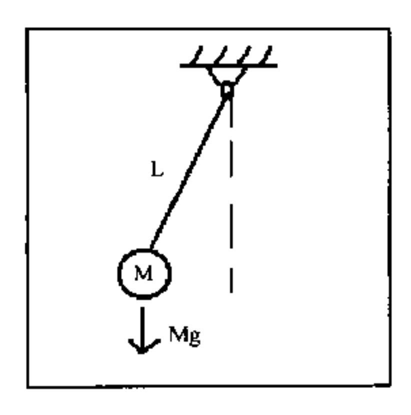


图 2-6 单摆系统示意图

及角速度与位移的关系图。

结果:系统的运动方程为:

$$ML\ddot{\theta} + BL\dot{\theta} + Mg\sin\theta = 0$$

系统的仿真结果为:

其横坐标为时间,纵坐标为单摆系统的角位移时间响应,如图 2-7 所示。从图 2-7 中的响应曲线的类型来看,是代阻尼的三角函数型,围绕原点左右振荡,并随时间的递增振荡幅度逐渐减小(这一点不太明显,不过仔细观察图 2-7 可以发现响应曲线第一个峰值约为 0.9,第二个峰值则约为 0.85,逐渐衰减)。

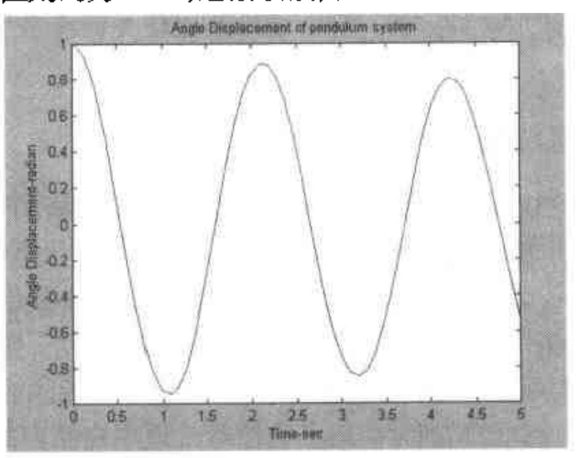


图 2-7 单摆系统位移响应曲线

其横坐标为时间,纵坐标为单摆系统的角速度时间响应,如图 2-8 所示。同样,响应曲线也是代阻尼的三角函数型,这是因为速度是位移的微分。

其横坐标为单摆系统的位移,纵坐标为角速度,如图 2-9 所示。

求解过程:

为得到上图的结果,在 MATLAB 环境下共分三步进行:

1. 编写系统微分方程

在 MATLAB Command Window 下键入"edit"或选择"File"菜单新建 M-file,进入 MATLAB Editor/Debugger,编辑 M 函数"pendulum":

```
function xt = pendulum(t,x)
g=9.81;
```

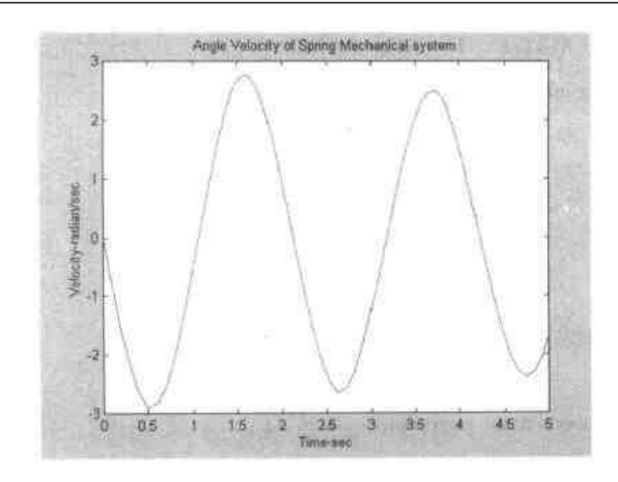


图 2-8 单摆系统速度响应曲线

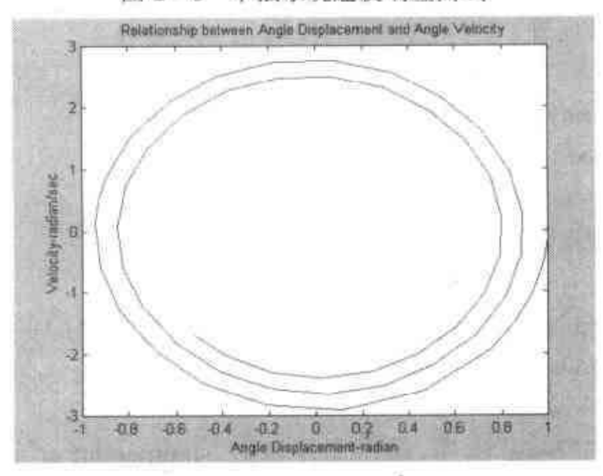


图 2-9 单摆系统位移与速度关系曲线

```
M = 1;
B = 0.1;
L = 1;
xt = [x(2); - B/M * x(2) - g/L * sin(x(1))];
```

选择 MATLAB Editor/Debugger 的"File"菜单的"Save"选项，保存文件名为"pendulum.m"。

2. 编写系统的仿真程序

选择"File"菜单新建 M-file，键入以下代码：

```
%关闭所有图形窗口
close all
%数据初始化
tfinal = 5;
%仿真开始和结束时间
tspan = [t0 tfinal];
%系统初始条件
x0 = [1,0];
%设置 ode113()函数的可选参数
```

```
options = odeset('AbsTol',[1e-6;1e-6]);
[t,x] = ode113('pendulum',tspan,x0,options);
%绘图
%位移向量
theta = x(:,1);
%速度向量
av = x(:,2);
%绘制时间 - 位移图
plot(t,theta);
title('Angle Displacement of pendulum system');
xlabel('Time - sec');
ylabel('Angle Displacement - radian');
%绘制时间 - 速度图
plot(t,av);
title('Angle Velocity of Spring Mechanical system');
xlabel('Time - sec');
ylabel('Velocity - radian/sec');
%绘制位移 - 速度图
plot(theta,av);
title('Relationship between Angle Displacement and Angle Velocity');
xlabel('Angle Displacement - radian');
ylabel('Velocity - radian/sec');
```

选择"File"菜单的"Save"选项,保存文件名为"simupendu.m",同样要注意和 pendulum.m 保存在同一路径下。

3.运行

选择"Tools"菜单的"Run"选项或在 MATLAB Command Window 下直接键入文件名 simupendu,在 MATLAB Command Window 下查看运行的结果。

分析:与例 2.1 的弹簧机械系统相同,本例的系统运动方程也是二阶的微分方程。因此,首先要化为两个一阶的微分方程,然后再进行仿真。根据系统的运动方程:

$$ML\ddot{\theta} + ML\dot{\theta} + Mg\sin\theta = 0$$

令:

$$x_1 = \theta, x_2 = \dot{\theta}$$

有:

$$\frac{dx_1}{dt} = x_2$$

$$\frac{dx_2}{dt} = -\frac{B}{M}x_2 - \frac{g}{L}\sin x_1$$

这两个一阶微分方程就是系统的状态方程,这里选择的两个状态分别是单摆的角位移和角速度。

从仿真结果来分析,单摆的角位移和角速度都是过零的正弦曲线,符合事先对其受力的分析。从图 2-9 可以看出,系统的角位移与角速度的关系曲线是一支内旋的螺旋曲线,逐渐向原点方向旋转。这是由于阻尼系数 B 的存在,使得单摆的动能逐渐减小,最终动能为零,单摆回到原点。如果阻尼系数为零,我们将看到一个中心在原点的椭圆(或圆)。

事实上,类似图 2-9 这种系统的位移与速度曲线就是其相平面图,从图上可以判断系统的运动趋势、稳定性、有无平衡点等等一系列性质。当然,这种方法不如直接仿真来得精确和直观。

对于非线性系统,还可以按照 Taylor 展开的方法直接将其线性化,根据精度来控制展开的阶数。对于本例,设 $\theta=\theta_0+\Delta\theta$,则系统的运动方程如下:

$$ML(\ddot{\theta}_0+\Delta\ddot{\theta})+BL(\dot{\theta}_0+\Delta\dot{\theta})+Mg\sin(\theta_0+\Delta\theta)=0$$

当 $\Delta\theta$ 接近于零时,有 $\sin\Delta\theta\approx0,\cos\Delta\theta\approx1$;又因为单摆的振荡幅度较小,有 $\sin\theta_0\approx\Delta\theta$。因此,根据三角公式将系统的运动方程展开,有:

$$ML\Delta\ddot{\theta}+BL\Delta\dot{\theta}+Mg\Delta\theta=0$$

感兴趣的读者可以对这个线性微分方程进行仿真。通过仿真结果就会发现,在满足上文假设的条件时,其结果与根据非线性方程仿真的结果基本一致。当然,如果单摆的振荡幅度很大,上述的线性方程就不再成立,只能用非线性方程进行仿真了。

小结:本例通过对非线性的单摆系统进行仿真,简要介绍了对非线性系统进行数学描述的方法。事实上,读者可能已经发现了,线性系统仿真和非线性系统仿真在使用 ode113()函数的格式和用法上并无任何不同,区别仅仅在有关系统运动方程函数的编写上。因此,在使用 MATLAB 进行控制系统设计和仿真时,在大多数情况下不必刻意区分线性还是非线性系统,实际操作起来都是一样的。

需要注意的是,一般说来,非线性系统对于仿真的精度是比较敏感的,尤其是在系统稳定性和抗干扰能力的问题上,有可能在不同的仿真精度下得到不同的结果。因此,对于实际接触非线性系统,首先要进行理论分析,选择合适的模型和精度,然后在用 MATLAB 进行仿真。

有关系统的微分方程描述就介绍到这里,它是控制系统数学描述的基础,在以后的论述中还要经常提及。

2.2 控制系统的传递函数描述

上一节中我们讨论的是在时间域内描述动态系统的方法,即直接以微分方程为工具进行研究的方法。从这一节开始,我们要叙述另一种数学描述方法。这种方法不是直接去求解和讨论微分方程本身,而使用拉普拉斯变换建立一种数学模型,称为传递函数;用传递函数来研究对象的运动。在这种方法中,自变量不是时间,而是拉普拉斯变换中的复数变量 s,称为复频率。所以这种建筑在拉普拉斯变换和传递函数基础上的描述方法又称为频率域方法。简单地说,时间域描述方法以时间 t 自变量,频率域描述方法以复频率 s 为自变量。

线性时不变系统(Linear Time Invariant System,简称为 LTI 系统)的传递函数的定义为零初值条件下输出量的拉普拉斯变换与输入量的拉普拉斯变换像函数之比。尽管传递函数只能用于线性系统,但它比微分方程提供更为直观的信息。令传递函数的分母多项式的为零,便得到系统的特征方程。特征方程的根是系统的极点,分子多项式的根是系统的零点。那么传递函数便可由常数项与系统的零、极点决定。利用传递函数,我们可以方便地研究系统参数的改变对系统响应的影响。通过拉普拉斯反变换可以得到系统的时域

响应,这通常需要用到函数的部分分式展开。

通常有两种方法可以得到系统的传递函数。一种是根据定义,首先列写系统的微分方程,然后对该方程两边同时进行拉普拉斯变换,通过输出量拉普拉斯变换与输入量拉普拉斯变换像函数之比得到系统的传递函数。如果系统是线性的,那么传递函数就不因输入量函数或输出量函数的改变而改变。另一种是实验的方法,首先大致估计一下系统的阶次,然后加典型的测试信号,根据系统的响应曲线来求得相关的参数,最终获得系统的传递函数模型。这种方法一般适用于系统的机理比较难于分析,并且阶次不高的情况。当然,用这种方法所求得模型的误差也是比较大的。限于篇幅,本书所讨论的主要是第一种方法。在以下的叙述中,如果不作特殊说明,均假设系统的微分方程已知。

MATLAB 提供了许多功能强大的内部函数可以构建和处理系统的传递函数。包括多项式求根、求解传递函数的零极点和增益、传递函数部分分式展开等等。能够非常方便快捷地建立起系统的传递函数模型。

2.2.1 传递函数的零点和极点

设某控制系统只有一个输入量 $u(t)$,只有一个输出量 $y(t)$。并设这个系统可以用线性微分方程:

$$a_n \frac{d^n y}{dt^n} + a_{n-1}\frac{d^{n-1}y}{dt^{n-1}} + \cdots + a_1\frac{dy}{dt} + a_0 y =$$

$$b_m \frac{d^m u}{dt^m} + b_{m-1}\frac{d^{m-1}u}{dt^{m-1}} + \cdots + b_1\frac{du}{dt} + b_0 u$$

描述,其中 $n\geqslant 1, m\geqslant 1$,多项式首项不为零。假设 $u(t)$ 和 $y(t)$ 及其各阶导数在零负时刻的初值均为零。对上式两边同时取拉普拉斯变换,有:

$$a_n s^n \bar{y}(s) + a_{n-1}s^{n-1}\bar{y}(s) + \cdots + a_1 s\bar{y}(s) + a_0 y(s) =$$

$$b_m s^m u(s) + b_{m-1}s^{m-1}\bar{u}(s) + \cdots + b_1 s\bar{u}(s) + bu(s)$$

令:

$$G(s) = \frac{\bar{y}(s)}{u(s)}$$

有:

$$G(s) = \frac{b_m s^m + b_{m-1}s^{m-1} + \cdots + b_1 s + b_0}{a_n s^n + a_{n-1}s^{n-1} + \cdots + a_1 s + a_0}$$

此 $G(s)$ 就称为系统的传递函数。这也就是上文提到的第一种由微分方程得到系统传递函数的方法。

从纯数学的角度来看,传递函数就是两个有理多项式之比。传递函数的分母多项式称为系统的特征多项式,此多项式的根称为系统的极点。传递函数的分子多项式的根称为系统的零点。对于 LTI 系统来说,系统的所有信息都包含在其传递函数的极点和零点之中。研究传递函数,首先就要研究其极点和零点。

对于上文提到的只有一个输入量和一个输出量的系统,称为 SISO 系统(Single - Input Single - Output)。在 MATLAB 环境下可以方便地用两个行向量表示 SISO 系统的传递函数,分别代表其分子和分母的系数:

$$num = (b_m, b_{m-1}, \cdots, b_1, b_0) \qquad den = (a_m, a_{m-1}, \cdots, a_1, a_0)$$

两个行向量的元素分别是原 LTI 系统传递函数分子和分母多项式系数的降幂排列。

这两个行向量可以取不同的名字。但在 MATLAB 环境下对控制系统进行仿真时，习惯用 num 和 den 来命名，这样可读性较强。

有了描述系统传递函数的行向量之后，就可以求取其零点和极点了。MATLAB 提供了一条 tf2zp()函数，可以用来根据系统的传递函数求取其零点、极点和增益。其基本调用格式如下：

(Z,p,k) = TF2ZP(NUM,den)

其中 Z 和 p 是行向量(对于一阶系统来说是标量)，包含了系统的零点和极点。k 是标量，表示系统的增益。也就是说，tf2zp()函数将原系统的传递函数化为如下的形式：

$$G(s) = k\frac{(s-z_1)(s-z_2)\cdots(s-z_m)}{(s-p_1)(s-p_2)\cdots(s-p_n)}$$

其中　　$Z=(z_1,z_2,\cdots,z_m)$，$p=(p_1,p_2,p_n)$

根据系统的传递函数，令 $s=j\omega$，还可以求取系统的频率响应，从而对系统的性能作出评价。MATLAB 提供了一条函数 freqs()，可以根据系统的传递函数求取其频率响应数据。其基本调用格式为：

H = FREQS(B,A,W)

其中 B 和 A 是系统传递函数的分子和分母多项式，W 是频率采样点；H 是行向量，其元素是对应采样点 W 的系统复频率响应。下面通过一个简单的例子，简要介绍一下这些函数的基本用法。

[例 2.3]　考虑前面例子中的弹簧机械系统，其系统运动方程如下：

$$M\frac{d^2x}{dt^2}+B\frac{dx}{dt}+Kx=f(t)$$

其中 M = 1kg，K = 20N/m，B = 5N/m/s。仅在 t = 0 时，施加外力 f(t) = 1N。试给出系统的传递函数，求解其零点、极点和增益，并绘制其复平面直角坐标的 Nyquist 图和系统的频率响应曲线。

结果：系统的传递函数为：　$G(s)=\frac{1}{s^2+5s+20}$

系统的零点、极点和增益分别为：

```
system zero - points are z = Empty matrix: 0 - by - 1
system polar - points are
p = -2.5000 + 3.7081i -2.5000 - 3.7081i
system gain is k = 1
```

系统仿真结果如图 2-10 所示。

其横坐标为系统频率响应的实部，纵坐标为系统频率响应的虚部，如图 2-11 所示。

分析：本例的主要目的是分析系统传递函数本身的性质，因此取输入函数 $f(t)=\delta(t)$ 当然，系统的频率特性曲线的形状和稳定性与输入量的选取无关。从 tf2zp()函数输出的结果来看，系统没有零点，这与原传递函数分子多项式为常数项是相吻合的。系统有两个极点 -2.5+3.7801i 和 -2.5-3.7801i，实部小于零，极点均在复平面的左半平面，说明系统是稳定的。从系统的复平面直角坐标 Nyquist 图来看，系统的 Nyquist 曲线并不包围(-1,0)点。右半平面零点数为零，根据 Nyquist 稳定判据，系统是稳定的。这同前边对系统极点的分析得出的结果是一致的。

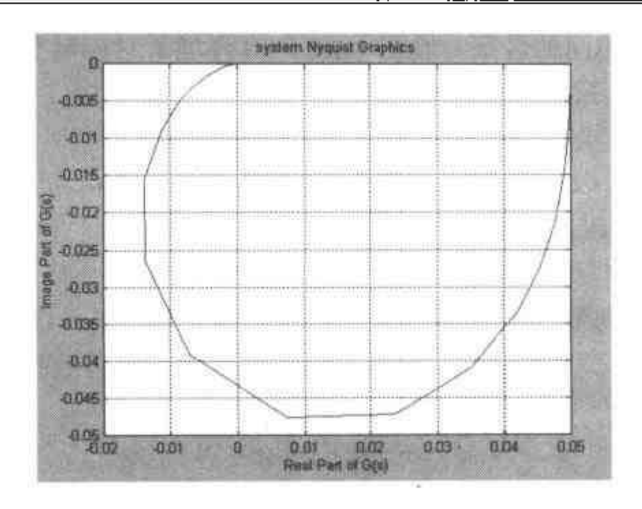


图 2-10 弹簧机械系统复平面直角坐标 Nyquist 曲线

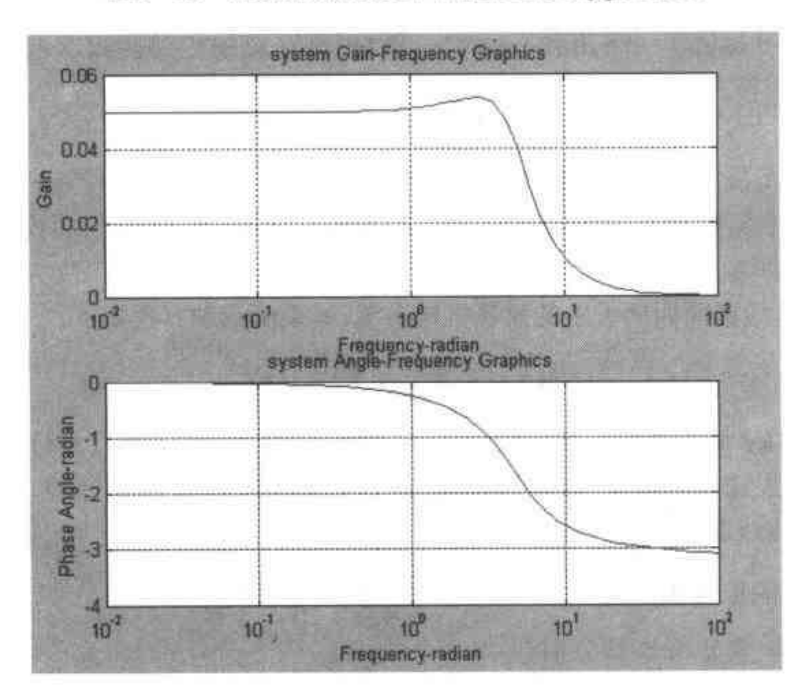


图 2-11 弹簧机械系统频率响应曲线

对照例 2.3 中系统的时间响应曲线，系统最终将趋于一个平衡点，因此系统是稳定的。系统的 Nyquist 曲线是半个心脏线的形状，这是因为对频率 ω 采样的过程中只选取了$(0,+\infty)$的点。如果包含$(-\infty,0)$的频率点，那么系统的 Nyquist 曲线应该是一个完整的心脏线的形状。不过对于所有的 LTI 系统来说，其 Nyquist 曲线均关于实轴对称，仅选取$(0,+\infty)$的频率点就足以看出曲线的形状和变化趋势了。

在绘制系统的频率响应曲线时使用了 x 轴的半对数坐标系，横轴是对数频率，纵轴分别是传递函数频率响应的模和相角。从系统的幅频响应曲线来看，大约在 $\omega=4$ 左右的地方有一个谐振峰。谐振峰越大，系统的响应曲线振荡约剧烈。也就是说，系统对该频率的输入信号响应最为敏感。本系统的谐振峰比较小，说明系统的运动比较平缓。从例2.3 系统的时间响应曲线来看，只经过一次振荡就趋于稳态值，验证了上面的说法。从系统的相频响应曲线来看，相角趋于 $-180°(-\pi)$ 时传递函数的模已趋于 0，与 1 相差很远。因

此,系统有较大的稳定裕量,存在充分的空间可以增加控制手段,使其输出量跟随期望的设定值。

求解过程:

本例的解题步骤分为三部分。

1. 求取系统的传递函数

对系统的运动方程两边取拉普拉斯变换,得:

$$Ms^2\bar{x}(s) + Bs\,\bar{x}(s) + K\,x(s) = f(s)$$

$$G(s) = \frac{1}{Ms^2 + Bs + K} = \frac{1}{s^2 + 5s + 20}$$

2. 求解系统的零点、极点和增益并绘制系统频率响应曲线

在 MATLAB Command Window 下键入"edit"或选择"File"菜单新建 M - file,进入 MATLAB Editor/Debugger,编辑 M 文件"springfreq.m":

```
%关闭所有图形窗口
close all;
%数据初始化
%传递函数分子多项式
num = 1;
%传递函数分母多项式
den = [1 5 20];
%求系统零点、极点和增益
w = logspace(-2,2);
[z,p,k] = tf2zp(num,den);
%显示结果
disp('system zero - point is');
%系统零点
z
disp('system polar - point is');
%系统极点
p
disp('system gain is');
%系统增益
k
%求系统的频率响应
H = freqs(num,den,w);
x = real(H);
y = imag(H);
%图形绘制
%复平面直角坐标 Nyquist 图
plot(x,y);
title('system Nyquist Graphics');
xlabel('Real Part of G(s)');
ylabel('Image Part of G(s)');
```

```
grid;
subplot(211),
%传递函数增益
q = abs(H);
semilogx(w,q);
title('system Gain - Frequency Graphics');
xlabel('Frequency - radian');
ylabel('Gain');
grid;
%传递函数相角
alpha = angle(H);
subplot(212),
semilogx(w,alpha);
title('system Angle - Frequency Graphics');
xlabel('Frequency - radian');
ylabel('Phase Angle - radian');
grid;
```

选择“File”菜单的“Save”选项，保存文件名为“springfreq.m”。

3. 运行

选择“Tools”菜单的“Run”选项或在 MATLAB Command Window 下直接键入文件名 springfreq，在 MATLAB Command Window 下查看运行的结果。

小结：本例从频率域的角度重新分析了例 2.1 介绍的弹簧机械系统，所得到的结果与原先的结论是完全一致的。

应该说，时间域分析方法和频率域分析方法各有所长，各自有其适用的领域。时间域分析方法的特点是非常直观，所有想要的信息一目了然，分析结果与直接应用紧密结合，并且适用与各种线性和非线性系统。

频率域分析虽然是间接的方法，但对于线性系统分析来说，所有信息都包含在其传递函数之中了。并且，人们总结了大量关于频率域分析的规律和定理，直接应用这些规律和定理比时间域分析方法方便快捷，能受到事半功倍的效果。例如判断系统的稳定性时，只需求解系统极点的实部即可，不必绘制其时间响应曲线

对于控制领域的专业人员来说，虽然数字式计算机包括类似 MATLAB 等仿真软件的出现，使得时间域分析同样简便易行，但同时熟练掌握这两种分析方法还是非常必要的。它们互为表里，相互补充。

在上文的程序中使用了一个以前没有接触过的函数 logspace()，其调用格式为 LOGSPACE(d1, d2, N)，功能是产生 N 个对数等间距的采样点，范围是 10 的 d1 次幂到 10 的 d2 次幂之间。如果不给出参数 N，则产生 50 个对数等间距的采样点。细心的读者可以发现，因为对数函数的限制，这里的采样点均大于零。事实上，这也是为什么上文系统的复平面直角坐标 Nyquist 图只绘制了实轴以下部分的原因。

对于离数控制系统来说，其运动方程是以差分方程的形式描述的。相应的对其差分方程两边同时进行 Z 变换，就可以得到该离数控制系统的脉冲传递函数。脉冲传递函数

在 MATLAB 中的实现方法和处理手段与 s 域的传递函数基本一致，也是用两个行向量来代表脉冲传递函数的分子和分母多项式。只是在分析过程中要注意到函数的自变量由 s 域转化为 Z 域，s 域的左半平面转化为 Z 域的单位圆内部。

传递函数还可以描述多输入多输出的系统（Multi - Input Multi - Output，简称 MIMO 系统），这就引出了传递函数矩阵的概念。传递函数矩阵的描述方法很多，英国学派的多变量频域设计方法就是基于系统的传递函数矩阵描述的。限于篇幅，对于脉冲传递函数和传递函数矩阵这里就不再介绍了，以后接触到相关的概念时会作一些初步的解释感兴趣的读者请参阅相关的资料。

2.2.2 传递函数的部分分式展开

根据传递函数系统的时间域响应时，通常要用到有理分式的部分分式展开 所谓部分分式展开，就是将高阶的有理分式化为若干个一阶有理分式之和的形式 如果传递函数 G(s)不包含多重极点，那么，将 G(s)用部分分式展开后即可得到：

$$G(s) = k + \sum_{i=1}^{n} \frac{r_i}{s - p_i}$$

k 是常数项，对于真分式来说 k = 0。r 是各分式的系数，p 是系统的极点。

对上式求拉普拉斯逆变换，可得系统的冲激响应为：

$$y(t) = k\delta(t) + \sum_{i=1}^{n} r_i e^{p_i t}$$

如果要求解系统在输入函数 u(s) 下的时间响应，只需对 G(s)·u(s)进行部分分式展开并求拉普拉斯逆变换即可。

MATLAB 提供了一条函数 residue()可以求解有理分式的部分分式展开，其基本调用格式为：

(R,P,K) = RESIDUE(B,A)

其中 B 和 A 分别表示降幂排列的该有理分式的分子和分母多项式系数；R 是求得的部分分式展开的各分式系数，P 是系统极点，K 是常数项。有了这些结果，就可以根据部分分式展开求解系统的时间响应了。请看例 2.4

[例 2.4] 根据例 2.3 的系统传递函数，试求取系统的部分分式展开，并绘制系统的冲激响应和阶跃响应。

系统传递函数为： $G(s) = \dfrac{1}{s^2 + 5s + 20}$

结果：传递函数的部分分式展开系数为：

```
system Fraction coefficients are
r = 0 - 0.1348i
    0 + 0.1348i
system polar - points are
p =   - 2.5000 + 3.7081i
      - 2.5000 - 3.7081i
system Direct Term is
k = [    ]
```

表达式为:

$$G(s)=\frac{-0.1348i}{s+2.5-3.7801i}+\frac{0.1348i}{s+2.5+3.7801i}$$

系统的冲激响应为:

其横坐标为时间,纵坐标为响应值,如图 2－12 所示。

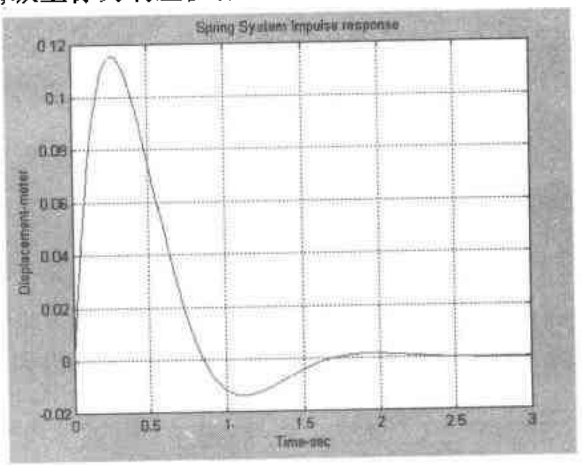


图 2－12 弹簧系统的冲激响应曲线

系统的阶跃响应曲线如图 2－13 所示。

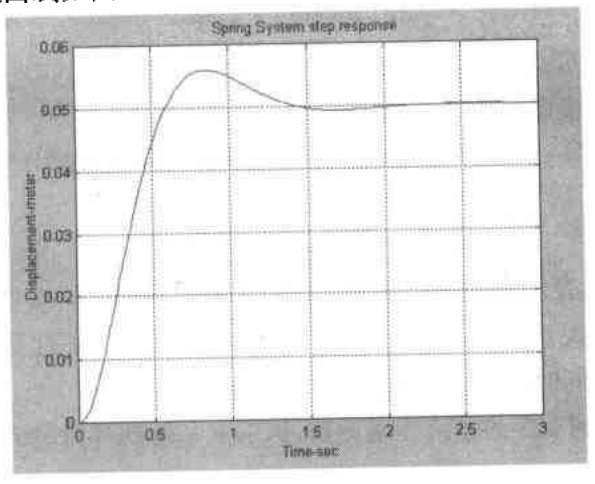


图 2－13 弹簧系统的阶跃响应曲线

分析:residue( )函数的返回值都是以向量的形式存在的。因此,"k = [ ]"表示 k 的值为零。从系统传递函数部分分式展开的结果来看,其极点均为复数。对于二阶系统来说,这表示系统的时间响应是振荡型的。系统的冲激响应就是输入函数为 δ(t)的系统时间响应。因为 δ(t)的拉普拉斯变换是 1,所以事实上就是系统传递函数的时间响应。

系统的时间域响应曲线是图 2－12,对应的频率域曲线是例 2.3 的图 2－11。从图 2－12 可以看出,系统的冲激响应在 T = 0.25s 左右达到峰值,对应于图 2－11 的 ω = 4 左右的谐振峰。系统的阶跃响应是输入函数为 1u(t)时系统的时间响应,其响应曲线是图 2－13。可以看出,图 2－13 与图 2－3 弹簧系统的位移时间响应曲线的形状完全一致,仅仅是纵坐标的刻度不同。事实上,图 2－3 是系统的输入函数为 30u(t)时的响应曲线,与

图 2－13 在纵坐标的刻度上相差 30 倍。细心的读者还会发现，图 2－12 与图 2－4 的曲线形状也完全一致，仅仅是纵坐标相差 30 倍。但这并不表明图 2－12 也是速度的时间响应曲线。这是因为图 2－4 是速度的时间响应曲线，速度是位移的导数，将该曲线积分就可以得到位移的时间响应曲线。而 δ(t) 正式 u(t) 的导数，对于 LTI 系统来说，导数和积分都是线性运算，是可以传递的。因此，将系统的冲激响应曲线积分就可以得到系统的阶跃响应曲线。所以，图 2－12 仍是位移的时间响应曲线。

求解过程：

本例的解题步骤分为两部分。

1．求取系统传递函数的部分分式展开和冲激响应

在 MATLAB Command Window 下键入“edit”或选择“File”菜单新建 M－file，进入 MATLAB Editor/Debugger，编辑 M 文件“springimp.m”：

```
%关闭所有图形窗口
close all;
%求取传递函数部分分式展开
num = 1;
den = [1 5 20];
[r p k] = residue(num,den);
disp('system Fraction coefficients are');
%各分式系数
r
disp('system polar - points are');
%系统极点
p
disp('system Direct Term is');
%常数项
k
%求解系统的时间响应
t = 0:0.01:3;
q = size(t);
m = length(k);
n = length(r);
s = zeros(q);
%判断是否有常数项
if m ~ = 0
    s(1) = k;
end
for i = 1:n
    s = s + r(i) * exp(p(i) * t);
end
%绘图
plot(t,s);
title('Spring System Impuls response');
```

```
xlabel('Time - sec');
ylabel('Displacement - meter');
grid;
```

选择"File"菜单的"Save"选项，保存文件名为"springimp.m"。选择"Tools"菜单的"Run"选项或在 MATLAB Command Window 下直接键入文件名 springimp，在 MATLAB Command Window 下查看运行的结果。

2. 求取系统的阶跃响应曲线

在 MATLAB Editor/Debugger 下，选择"File"菜单，"Open"选项，打开刚才编辑好的文件 springimp.m，对此 M 文件作下述修改：

（1）将第 5 行的 den = [1 5 20] 改为 den = [1 5 20 0]（行号可参看 MATLAB Editor/Debugger 右下角的"Line"。

（2）将第 27 行至第 29 行改为：

```
title('Spring System Impuls response');
xlabel('Time - sec');
ylabel('Displacement - meter');
```

选择"File"菜单的"Save as"选项，另存的文件名为"springstep.m"。选择"Tools"菜单的"Run"选项或在 MATLAB Command Window 下直接键入文件名 springstep，在 MATLAB Command Window 下查看运行的结果。

小结：本例从另一个角度再一次讨论了时间域分析方法和频率域分析方法之间的关系。利用部分分式展开和拉普拉斯逆变换的方法可以非常方便的将系统的频率域数据转化为时间域的数据。在展开的分式中，每一个分式就代表系统的一个运动模态。当然，在信号分析工具箱里 MATLAB 还提供了直接求解 s 函数拉普拉斯逆变换数值解的函数，但从概念上讲，还是用部分分式展开的方法更能反映控制系统的运动本质。本例编写的通过传递函数的部分分式展开求解系统时间响应的程序有一定的通用性，其核心代码能够求解不包含多重极点的各种传递函数的时间响应。读者可以将其改编为 M 函数，存在 MATLAB 的缺省路径下，方便以后的使用。程序中用到了 size() 和 length()，这分别是用来求取矩阵和向量大小的函数。其含义和功能都非常简单，读者查阅 MATLAB 帮助就可以理解和使用，这里就不再赘述了。

residue() 函数还可以根据部分分式展开求解原传递函数表达式，其调用格式为：

$$[B, A] = \mathrm{RESIDUE}(R, P, K)$$

各参数的含义与上文的解释完全一致。例如对于例 2.4 的结果，在 MATLAB Command Window 下直接键入：

```
r = 0 - 0.1348i
    0 + 0.1348i
p = -2.5000 + 3.7081i
    -2.5000 - 3.7081i
k = [ ]
[num,den] = residue(r,p,k)
num = 0 1
den = 1 5 20
```

可以看到,其结果与原传递函数完全相同。

相对于微分方程来说,控制系统的传递函数描述虽然是一种间接的方法,但其含义和数学基础同样是非常明确和坚实的。除了上文介绍的这些函数外,MATLAB 在复频域分析方面还有许多功能强大的函数,读者也可以编写 M 函数组建自己的工具箱,满足不同场合的特殊需要。

以上我们讨论的主要是 SISO 系统的数学描述,属于经典控制理论领域。下一节介绍的系统的状态方程描述是一种囊括了 SISO 系统和 MIMO 系统的数学描述方法,是所谓现代控制理论的基础

2.3 控制系统的状态方程描述

列出控制系统的原始运动方程组后,固然可以如上节所说的那样化成关于单一变量的高阶微分方程,但也可以化成另一种标准形式的方程组,就是状态方程组。用状态方程组描述动态系统,是现代时间域控制理论学派(状态空间学派)的一种基本手段。在用状态方程描述运动对象时,状态变量是最重要的基本概念之一。状态变量是这样来定义的:在描述对象运动的所有变量中,必定可以找到数目最少的一组变量,它们已经足以描述对象的全部运动。这组变量称为对象的状态变量。所谓足以描述系统的全部运动,是指:只要确定了这组变量在某一初始时刻 t = T 值,并且确定了从这一初始时刻起(t≥T)的输入量函数,则对象的全部变量在此刻和此后(t≥0)的运动都唯一确定了。

集总参数的线性网络可用微分方程表示为:

$$\dot{x}(t) = Ax(t) + Bu(t) \qquad y(t) = Cx(t) + Du(t)$$

该系统的一阶微分方程即为状态方程,X 是状态变量。状态空间方法易于采用数字和模拟计算机求解,MATLAB 的控制系统工具箱中也提供了大量功能强大的函数可以用来进行状态方程的求解和仿真。另外,状态空间方法容易拓展到非线性系统。有两种方法可以得到系统的状态方程。一种是从系统的 n 阶微分方程得到;另一种是从系统模型中选用合适的状态变量直接写出。本节主要讨论第一种方法,如果不作特殊说明,均假设系统的微分方程已知。

状态向量是状态空间控制理论的基本概念。在状态空间控制理论中使用状态方程来描述动态系统的运动。状态方程的主要特征是:在全部受控量中,只选择一组状态变量来列写方程,其他受控变量不进入方程;状态方程必须写成标准形式。本节将详细讨论的状态方程的数学描述,及其各种标准形式。

2.3.1 数学描述

假设一个 n 阶线性系统的微分方程描述如下所示。我们将讨论如何选取状态变量,得到该系统的状态方程描述,并给出状态方程在 MATLAB 环境下的实现方法。

$$a_n \frac{d^n x}{dt^n} + a_{n-1}\frac{d^{n-1}x}{dt^{n-1}} + \cdots + a_1 \frac{dx}{dt} + a_0 x$$

$$= b_n \frac{d^n u}{dt^n} + b_{n-1}\frac{d^{n-1}u}{dt^{n-1}} + \cdots + b_1 \frac{du}{dt} + b_0 u$$

其中 $x(t)$、$u(t)$分别是系统的输出量和输入量，$a_n \neq 0$。先计算列向量：

$$\begin{pmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \\ \vdots \\ \beta_n \end{pmatrix} = \begin{pmatrix} a_n & & & & \\ a_{n-1} & a_n & & 0 & \\ a_{n-2} & a_{n-1} & a_n & & \\ \vdots & \vdots & \vdots & \ddots & \\ a_0 & a_1 & a_2 & \cdots & a_n \end{pmatrix}^{-1} \begin{pmatrix} b_n \\ b_{n-1} \\ b_{n-2} \\ \vdots \\ b_0 \end{pmatrix}$$

然后用下列方程组替代系统的原微分方程：

$$\begin{pmatrix} \dot{\xi}_1 \\ \dot{\xi}_2 \\ \vdots \\ \dot{\xi}_{n-1} \\ \dot{\xi}_n \end{pmatrix} = \begin{pmatrix} & \vdots & & \\ 0 & \vdots & I_{n-1} & \\ \cdots & \cdots & \cdots & \cdots \\ -\frac{a_0}{a_n} & -\frac{a_1}{a_n} & \cdots & -\frac{a_{n-1}}{a_n} \end{pmatrix} \begin{pmatrix} \xi_1 \\ \xi_2 \\ \vdots \\ \xi_{n-1} \\ \xi_n \end{pmatrix} + \begin{pmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_{n-1} \\ \beta_n \end{pmatrix} u$$

$$x = \xi_1 + \beta_0 u$$

这样就用 n 个一阶微分方程和一个代数方程代替了 n 阶的原始微分方程，与此同时引入了 n 个新的受控量，连同 x，一共是 n + 1 个受控量。新方程均不含输入量 u 的导数项。转化为状态方程的标准形式，可以得到：

$$A = \begin{pmatrix} & \vdots & & \\ 0 & \vdots & I_{n-1} & \\ \cdots & \cdots & \cdots & \cdots \\ -\frac{a_0}{a_n} & -\frac{a_1}{a_n} & \cdots & -\frac{a_{n-1}}{a_n} \end{pmatrix} \qquad B = \begin{pmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_{n-1} \\ \beta_n \end{pmatrix}$$

$$C = (1 \quad 0 \cdots 0)_{1 \times n} \qquad D = \beta_0$$

这是方程右边包含输入量导数项的情况。而不包含输入量导数项的情况则比较简单，只需选择输出量 x 的各阶导数作为 n 个状态变量即可，这里就不再赘述了。

与传递函数的表示方法类似，在 MATLAB 环境下可以很直观地将系统的状态方程模型表示为 4 个不同维数的矩阵(A,B,C,D)，然后就可以使用 MATLAB 提供的控制系统工具箱中的各种函数对该系统操作了。请看下面这个著名的例子(Automatic Control Systems，作者 B. C. Kuo，第七版)。

[例 2.5] 如图 2-14 所示的电磁控制系统，磁性物质做成的小球在线圈磁力和自身重力的作用下产生运动。设小球质量 M = 0.05 kg，磁力系数 K = 0.0001，电感 L = 0.01 H，电阻 R = 1 Ω，重力加速度 g = 9.81 m/s^2；V 是电源电压，i 是线圈中电流，b 是小球与线圈的距离。根据系统的微分方程，试选取合适的状态变量，给出系统的状态方程，并讨论 h = 0.01m，V = 1V 时系统的运动情况。

结果：系统的微分方程为：

$$M\frac{d^2h}{dt^2} + K\frac{i^2}{h} - Mg = 0$$

$$L\frac{di}{dt} + iR - V = 0$$

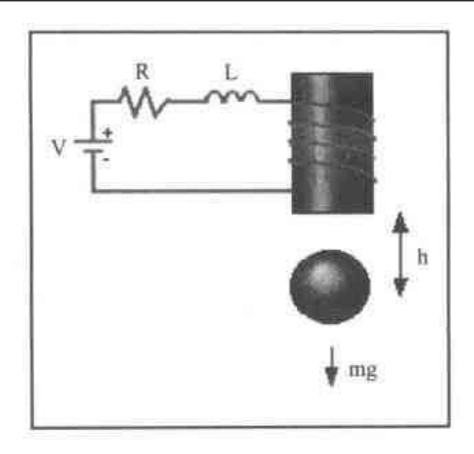


图 2-14 电磁控制系统示意图

选择系统的状态变量为 $X=[\Delta h \quad \Delta\dot{h} \quad \Delta i]^T$，输出量为 $y=h$，输入量 $u=V$，可得系统的状态方程为：

$$\dot{X}=\begin{pmatrix}0 & 1 & 0\\ 980 & 0 & -2.8\\ 0 & 0 & 100\end{pmatrix}X+\begin{pmatrix}0\\ 0\\ 100\end{pmatrix}u$$

$$y=[1 \quad 0 \quad 0]X$$

系统的开环时间响应，如图 2-15 所示。

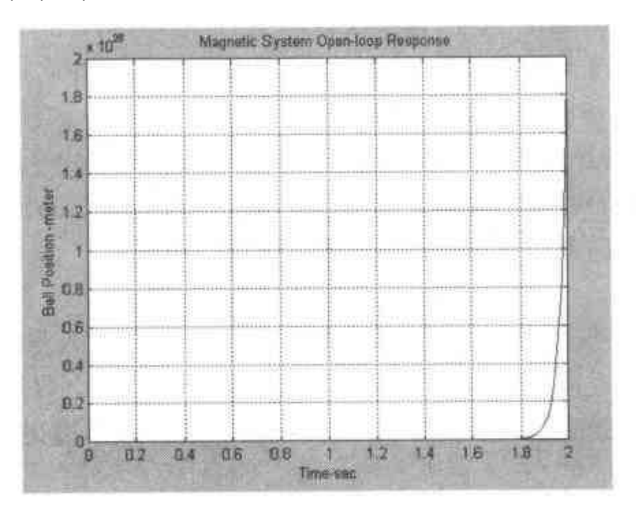


图 2-15 电磁控制系统开环响应

分析：本例系统的微分方程是不包含输入量导数项的情况，因而可以直接选取状态变量。本例选取的状态变量是距离 Δh、Δh 的一阶导数和电流 Δi，这是最一般的思路。因为所选的量都是相对变化量，所以加 Δ 号作标记。从图 2-15 系统的开环时间响应来看，相对距离 Δh 从零开始逐渐增大，直至 +∞。这是因为系统的磁力不足以抵销小球的重力，小球在重力加速度的作用下逐渐远离带电线圈的缘故。根据图 2-15 的信息，我们可以判定系统的开环响应是不稳定的。

求解过程：

在 MATLAB Command Window 下键入“edit”或选择“File”菜单新建 M-file，进入

MATLAB Editor/Debugger，编辑 M 文件“magneticsys.m”：

```
%关闭所有图形窗口
close all;
%状态方程模型
A=[0 1 0;980 0 -2.8;0 0 -100];
B=[0;0;100];
C=[1 0 0];
D=0;
%仿真时间初始化
t = 0:0.01:0.2;
u = zeros(size(t))+1;
x0 = [0.01 0 0];
[y,x] = lsim(A,B,C,0,u,t,x0);
h = x(:,1);
%绘图
plot(t,h)
title('Magnetic System Open-loop Response');
xlabel('Time-sec');
ylabel('Ball Position -meter');
grid;
```

选择“File”菜单的“Save”选项，保存文件名为“magneticsys.m”。选择“Tools”菜单的“Run”选项或在 MATLAB Command Window 下直接键入文件名 magneticsys，在 MATLAB Command Window 下查看运行的结果。

小结：本例简要介绍了选择状态变量和求取系统状态方程的方法，并给出了状态方程在 MATLAB 环境下的表示方法和处理手段。程序中用到了 lsim()函数，其功能是仿真 LTI 系统强迫输入的时间响应。该函数的基本调用格式为：

(Y,X) = LSIM(SYS,U,T,X0...)

其中 SYS 就是系统的状态方程参数(A,B,C,D)，U 是输入量，T 是仿真时间向量，X0 是系统初值；Y 是返回的输出量，X 是返回的状态变量的历史数据。

对于离散控制系统来说，也可以用状态方程对其进行数学描述。将其差分方程化为状态方程模型后，有：

$$\begin{cases} x(k+1) = Gx(k) + Hu(k) \\ y(k) = Cx(k) + Du(k) \end{cases}$$

简记为(G,H,C,D)。在 MATLAB 环境下，其表示方法和处理手段与连续 LTI 系统并无区别。事实上，使用数字式计算机进行仿真时，连续系统也要先进行采样量化，化为离散系统，然后再进行处理。因此，从实际操作的角度来看，离散控制系统和连续系统并无多大区别。当然，在进行理论研究时，还是要注意将二者区分开来。

2.3.2 对角化与 Jordan 标准型

从本节开始，我们将讨论系统的状态方程模型的各种标准型。

给定 LTI 系统(A,B,C,D)，输入量 u(t)和状态变量的初值 x_0 后，可以求出其状态方

程的解为：

$$x(t)=e^{At}x_0+\int_0^t e^{A(t-\tau)}Bu(\tau)d\tau$$

要求解此方程，很自然就想到将矩阵 A 对角化，将其化为对角元为特征值的对角矩阵。这样一来，上述方程里的矩阵指数也就化成了对角元是 $e^{\lambda_1 t},e^{\lambda_2 t},\cdots,e^{\lambda_n t}$ 的对角矩阵（假定 A 有 n 个互不相同的特征值）。

对于线性系统（A，B，C，D）来说，我们可以找到非奇异变换矩阵 W，使得：

$$A'=W^{-1}AW=\begin{pmatrix}\lambda_1 & & & \\ & \lambda_2 & & \\ & & \ddots & \\ & & & \lambda_n\end{pmatrix}$$

$$B'=W^{-1}B \qquad C'=CW \qquad D'=D$$

此系统的新的表示形式（A'，B'，C'，D'）称为系统的对角规范型。W 是矩阵 A 的特征向量组成的矩阵。对应的状态变量变为 X = WX'。请看下例：

［例 2.6］ 给定某控制系统的状态空间描述为：

$$\dot{X}=\begin{pmatrix}0 & 1 & -1\\ -6 & -11 & 6\\ -6 & -11 & 5\end{pmatrix}X+\begin{pmatrix}0\\0\\1\end{pmatrix}u$$

$$y=(1\quad 0\quad 0)X$$

试求其对角规范型和变换矩阵 W，并根据其对角规范型绘制系统在初值为 X = [5;10;15]，u = 0 时的响应曲线。

结果：系统的对角规范型为：

```
The Diagonal Form of System is:
L  = -1.0000     -0.0000     -0.0000
      0.0000     -2.0000     -0.0000
     -0.0000      0.0000     -3.0000
b  = -2.8284
     -13.7477
      10.8628
c  =  0.7071     -0.2182     -0.0921
```

系统对角规范型的变换矩阵为

```
Transformation Matrix is:
W  = 0.7071     -0.2182     -0.0921
     0.0000     -0.4364     -0.5523
     0.7071     -0.8729     -0.8285
```

系统在 u = 0 时的响应曲线，如图 2－16 所示。

分析：从系统的响应曲线来看，各状态变量最终都趋于零。这一方面是因为输入量 u = 0，另一方面是因为初值不为零，但矩阵 A 的特征值均为负值，系统的零输入响应呈衰减趋势。但状态变量 x(3)下降幅度和速度最大，x(1)下降幅度和速度最小。这是因为状态变量 x(3)对应的特征值是－3，而 x(1)对应的特征值是－1，其衰减速率不同。

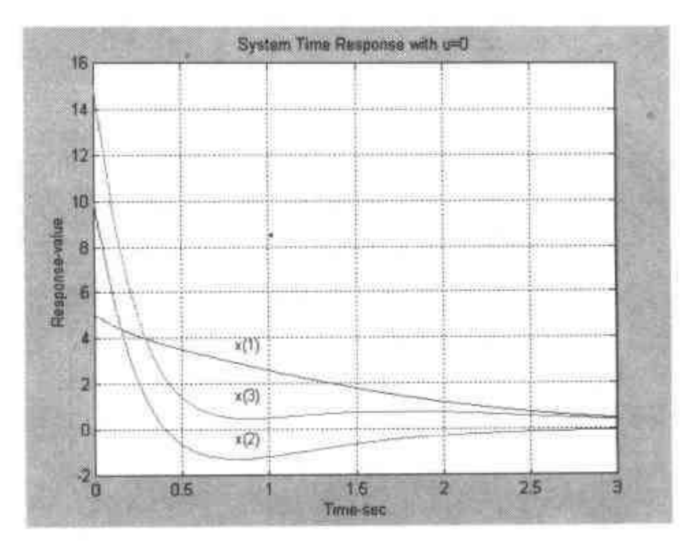


图 2-16 系统状态方程模型时间响应曲线

求解过程：

在 MATLAB Command Window 下键入"edit"或选择"File"菜单新建 M-file，进入 MATLAB Editor/Debugger，编辑 M 文件"diagform.m"：

```
%关闭所有图形窗口
close all;
%状态方程模型
A=[0 1 -1;-6 -11 6;-6 -11 5];
B=[0;0;1];
C=[1 0 0];
%求取对角规范型
[W,lamda]=eig(A);
L=inv(W)*A*W;
b=inv(W)*B;
c=C*W;
%显示结果
disp('The Diagonal Canonical Form of System is:');
L
b
c
disp('Transformation Matrix is:');
W
%仿真数据初始化
t=0:0.01:3;
x0=[5;10;15];
xx0=inv(W)*x0;
n=length(t);
x=zeros(3,n);
xx=zeros(3,n);
```

```
%求解状态变量
for i=1:n
xx(:,i)=expm(L*t(i))*xx0;
x(:,i)=W*xx(:,i);
end
%绘图
plot(t,x)
title('System Time Response with u=0');
xlabel('Time-sec');
ylabel('Response-value');
text(0.8,3.7,'x(1)');
text(0.8,1.5,'x(3)');
text(0.8,-0.4,'x(2)');
grid;
```

选择"File"菜单的"Save"选项，保存文件名为"diagform.m"。选择"Tools"菜单的"Run"选项或在 MATLAB Command Window 下直接键入文件名 diagform，在 MATLAB Command Window 下查看运行的结果。

小结：通过对角规范型求解系统状态方程的解的优点是，不必计算矩阵指数，只需计算单个的标量的指数即可。这样的好处是可以避免计算矩阵指数带来的算法的不稳定性。当然，由于不必手工计算，在 MATLAB 环境下这两种算法在程序编制的工作量上没有什么太大的差别，但在计算复杂性上还是有一定差别的。在本例具体求解的过程中，需要注意的是求得对角规范型的状态变量 xx 的时间响应之后，还要用变换矩阵 W 将其转化为原状态变量 x，然后再绘图。另外，如果变换矩阵 W 的向量排列的顺序不同，所求得的对角规范型在排列顺序上也稍有差别。

熟悉线性代数的读者都清楚，如果 $n\times n$ 矩阵 A 的线性无关特征向量个数小于 n 时，就无法将其对角化，也就无法将该线性系统化为对角规范型。在这种情况下，只能将其化为约当规范型(Jordan Canonical Form)。

$$A'=V^{-1}AV=J=\begin{pmatrix} J_1 & & & \\ & J_2 & & \\ & & \ddots & \\ & & & J_p \end{pmatrix}$$

设 A 矩阵有重特征值，其相异的特征值为 p 个。则必存在非奇异线性变换矩阵 V，将其化为约当型：

其中：
$$J_i=\begin{pmatrix} \lambda_i & 1 & 0 & \cdots & 0 & 0 & 0 \\ 0 & \lambda_i & 1 & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & 0 & 0 \\ 0 & 0 & 0 & \cdots & 0 & \lambda_i & 1 \\ 0 & 0 & 0 & \cdots & 0 & 0 & \lambda_i \end{pmatrix}$$

变换矩阵 V 是由 A 矩阵对应特征值的广义特征向量组成的。即：

$$V = (V_1 \quad \vdots \quad V_2 \quad \vdots \quad \cdots \quad \vdots \quad V_p)$$

$$AV_i = V_i J_i, (i = 1, 2, \cdots, p)$$

与对角规范型相似，B′矩阵和 C′矩阵也可通过变换矩阵 V 得到，只需将表达式中的 W 换成 V 即可。MATLAB 控制系统工具箱提供了一个求取矩阵 A 规范型的函数 jordan()，其基本调用格式为：

[V,J] = JORDAN(A)

其中 V 是变换矩阵，其列向量是矩阵 A 的广义特征向量。J 矩阵是求得的约当规范型。其具体用法请看下面这个简单的例子。

[例 2.7] 给定某控制系统的状态方程模型如下，试求其约当规范型。

$$\dot{X} = \begin{pmatrix} 4 & 6 & 0 \\ -3 & -5 & 0 \\ -3 & 6 & 1 \end{pmatrix} X + \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix} u$$

$$y = (1 \quad 0 \quad 0) X$$

结果：系统的约当规范型及其变换矩阵为：

```
The Jordan Canonical Form of System is
J =  -2     0     0
      0     1     1
      0     0     1
b =   5.0000
     -0.7500
      3.0000
c = -1     0     2
The Transformation Matrix is
V =  -1      0      2
      1      0     -1
     -3    -12      3
```

分析：从结果来看，系统矩阵 A 的约当标准型有两个约当块。第一个约当块阶次为 1，对应变换矩阵 V 中第一列列向量；第二个约当块阶次为 2，对应变换矩阵 V 中第二和第三列列向量，这两列向量也称为矩阵 A 的广义特征向量。根据前边介绍的 jordan()函数，可以求出这些值。

求解过程：

在 MATLAB Command Window 下键入“edit”或选择“File”菜单新建 M-file，进入 MATLAB Editor/Debugger，编辑 M 文件“jordanform.m”：

```
%关闭所有图形窗口
close all;
%状态方程模型
A = [4 6 0; -3 -5 0; -3 6 1];
B = [1;2;3];
C = [1 0 0];
```

```
D=0;
%求取系统的约当规范型
[V,J]=jordan(A);
b=inv(V)*B;
c=C*V;
%显示结果
disp('The Jordan Canonical Form of System are');
J
b
c
disp('The Transformation Matrix is');
V
```

选择"File"菜单的"Save"选项,保存文件名为"jordanform.m"。选择"Tools"菜单的"Run"选项或在 MATLAB Command Window 下直接键入文件名 jordanform,在 MATLAB Command Window 下查看运行的结果。

小结:jordan()函数对输入矩阵 A 的限制比较严格,要求矩阵 A 的元素必须是整数或整数之比。如果不满足这个条件的话,求得的约当规范型将与其真实值相差很远。另外,MATLAB 还提供了一条函数 canon(),也可以用来求解系统的约当规范型和伴随规范型,其基本调用格式为:

[CSYS,T] = CANON(SYS,TYPE)

其中 SYS 表示系统的原状态方程参数(A,B,C,D);TYPE 表示希望求得规范型的类型,其参数取值为:'modal' 代表约当规范型,'companion'代表伴随规范型;CSYS 表示求得的规范型参数(A',B',C',D');T 表示变换矩阵。

但这条函数有一个缺憾,就是 A 矩阵在无法完全对角化时有可能得出错误的结果。虽然此时 MATLAB 会给出警告,但我们还是建议在求解系统的约当规范型时,使用 jordan()函数。

2.3.3 可控规范型

状态可控性的含义是系统控制输入 u(t)支配状态变量 X(t)的能力。简单来说,就是系统的状态变量 X(t)能否在输入量 u(t)的控制下,从 n 维空间的任意一点出发,达到指定的 n 维空间的某一点。

某一状态变量 x(t)满足上述要求,则称该状态变量是可控的。若全体状态变量 X(t)均满足要求,则称该系统是完全可控的。

通过简单的推导,可得出线性定常系统状态可控性的代数判据为:

线性定常系统(A,B,C,D)状态完全可控的充分必要条件是,系统的可控性矩阵的秩为 n

$$Q_c = (B \quad \vdots \quad AB \quad \vdots \quad \cdots \quad \vdots \quad A^{n-1}B)$$

如果系统是状态完全可控的,则可以从可控性矩阵中挑出 n 个线性无关的列向量,以它们或它们的线性组合作为变换矩阵,就可以导出系统的第一可控规范型。系统的第一可控规范型为:

$$
A_{c1}=\begin{pmatrix}
 & & & & \vdots & -a_n \\
\cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\
1 & & & & \vdots & -a_{n-1} \\
 & 1 & & & \vdots & -a_{n-2} \\
 & & \ddots & & \vdots & \vdots \\
 & & & 1 & \vdots & -a_1
\end{pmatrix}
\qquad
B_{c1}=\begin{pmatrix} 1 \\ \cdots \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}
$$

$$
C_{c1}=(CB \;\vdots\; CAB \;\vdots\; \cdots \;\vdots\; CA^{n-1}B)
$$

其中：
$$
\begin{cases} B=Q_cB_{c1} \\ AQ_c=Q_cA_{c1} \\ CQ_c=C_{c1} \end{cases}
$$

这里假设 B 是 n 维列向量，则取变换矩阵就是系统可控性矩阵本身。此单输入系统的第一可控规范型的方框图如图 2－17 所示。

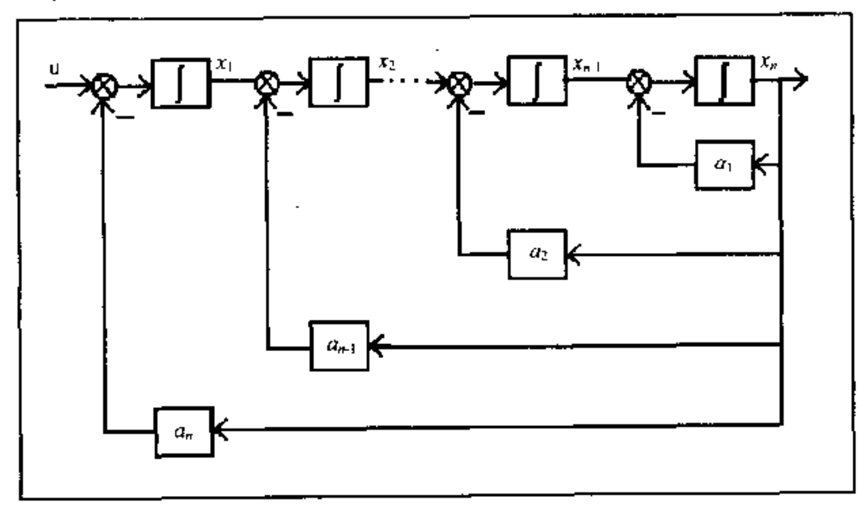


图 2－17　系统第一可控规范型方框图

图中的主通道是积分器串联形式，状态变量是每个积分器的输出，反馈通道是第一可控规范型矩阵 A 的最后一列向量。从图中可以看出，控制输入量 u 位于积分器串联链的最前方，因此，在输入端 u 加控制信号，可以控制系统的所有状态变量。这也是系统可控性的基本含义。

除系统的第一可控规范型之外，控制系统中常用的还有第二可控规范型等其他一些可控规范型。

这些规范型的形式和求解方法与第一可控规范型大同小异，都是首先根据可控性矩阵的某种线性变换来求解其变换矩阵，然后再求解可控规范型本身。因此，在以下的讨论中，我们一般只考虑第一可控规范型的情况。如果不特别指明，本书中的可控规范型都是指系统的第一可控规范型。

虽然 MATLAB 环境没有直接提供求解系统可控规范型的函数，但依靠本书第一章介绍过的有关矩阵运算的函数，可以很轻松地实现原系统与其可控规范型之间的相互转换。请看下面这个简单的例子：

[例 2.8] 给定线性定常系统状态方程模型如下，试判断系统的可控性。如果系统状态完全可控，试求其可控规范型和变换矩阵。

$$A=\begin{pmatrix}2 & 0 & 0 & 1\\0 & 4 & 1 & 3\\0 & 0 & 4 & 1\\0 & 0 & 0 & 2\end{pmatrix}\qquad B=\begin{pmatrix}1\\0\\1\\2\end{pmatrix}$$

$$C=\begin{pmatrix}1 & 1 & 0 & 0\end{pmatrix}$$

结果：

系统完全可控，其可控标准型为：

```
System is Controllable.
System First Controllable Canonical Form is:
Ac1 =          0         0          0      - 64.0000
          1.0000         0     - 0.0000      96.0000
        - 0.0000    1.0000          0      - 52.0000
          0.0000         0     1.0000        12.0000
Bc1  = 1
       0
       0
       0
Cc1  = 1 11 58 268
The Transformation Matrix is:
Q =   1    4    12    32
      0    7    46   236
      1    6    28   120
      2    4     8    16
```

求解过程：

在 MATLAB Command Window 下键入"edit"或选择"File"菜单新建 M - file，进入 MATLAB Editor/Debugger，编辑 M 文件"controlform.m"：

```
%系统状态方程模型
A=[2 0 0 1;0 4 1 3;0 0 4 1;0 0 0 2];
B=[1;0;1;2];
C=[1 1 0 0];
%系统阶次
n=length(A);
%求解系统可控性矩阵
Q=zeros(n);
Q(:,1)=B;
for i=2:n
%系统可控性矩阵的列向量
    Q(:,i)=A*Q(:,i-1);
end
%系统可控性矩阵的秩
```

```
m = rank(Q);
%判断系统是否状态完全可控,并求解可控规范型
if m = = n
%系统状态完全可控
    Acl = inv(Q) * A * Q;
    Bcl = inv(Q) * B;
    Ccl = C * Q;
%显示结果
    disp('System is Controllable.');
    disp('System First Controllable Canonnical Form is:');
    Acl
    Bcl
    Ccl
    disp('The Transformation Matrix is:');
    Q
else
%系统状态不完全可控
    disp('System State Variables cannot be totally controlled');
    disp('The rank of System Controllable Matrix is:');
%可控的状态变量数
    m
end
```

选择"File"菜单的"Save"选项,保存文件名为"controlform.m"。选择"Tools"菜单的"Run"选项或在 MATLAB Command Window 下直接键入文件名 controlform,在 MATLAB Command Window 下查看运行的结果。

小结:本例的核心代码可以作为一个求解系统第一可控规范型的 M 函数单独使用。当然,这个算法仅仅考虑了最简单的情况,并且没有编写注释。程序中使用了 rank()函数来判断系统可控性矩阵 Q 是否满秩,这牵扯到一系列病态条件讨论和算法稳定性的问题,这里就不再介绍了,感兴趣的读者请参阅相关的线性代数文献。

2.3.4 可观规范型

状态可观性的含义是系统输出量 y(t)反映状态变量 X(t)的能力。它回答了当系统输入量 u(t)=0 时能否通过输出量 y(t)的量测值确定状态变量 X(t)的问题。想要精确定义系统的完全可观,首先要定义所谓系统的不可观测状态:在有限的时间区间[T,T']内,把状态空间某个非零的有限点 x'作为系统的初态,x(T)=x',产生的时间响应在该时间区间内恒有 y(t)≡0,则称状态 x'为系统在时间区间[T,T']上的不可观测状态。如果状态空间中不存在不可观测状态,则称系统是在[T,T']上状态完全可观测的。

与系统的可控性相似,通过简单地推导,可得出线性定常系统状态可控性的代数判据为:

线性定常系统(A,B,C,D)状态完全可观的充分必要条件是,系统的可观性矩阵的秩为 n。

$$Q_o = (C \ \vdots \ CA \ \vdots \ \cdots \ \vdots \ CA^{n-1})^T$$

如果系统是状态完全可观的，则可以从可观性矩阵中挑出 n 个线性无关的列向量，以它们或它们的线性组合作为变换矩阵的逆，就可以导出系统的第一可观规范型。系统的第一可观规范型为：

$$A_{01} = \begin{pmatrix} & \vdots & & & \\ & \vdots & & I_{n-1} & \\ & \vdots & & & \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ -a_n & \vdots & -a_{n-1} & \cdots & -a_1 \end{pmatrix} \qquad B_{01} = \begin{pmatrix} CB \\ \cdots \\ \vdots \\ \cdots \\ CA^{n-1}B \end{pmatrix}$$

$C = (1 \ \vdots \ 0 \ \cdots \ 0)$ 其中变换矩阵：

$$P^{-1} = Q_o = (C \ \vdots \ CA \ \vdots \ \cdots \ \vdots \ CA^{n-1})^T, \begin{cases} B = PB_{01} \\ AP = PA_{01} \\ CP = C_{01} \end{cases}$$

状态完全可观系统的方框图与状态完全可控系统的方框图基本一致，也是由积分器串联链构成。只是串联链的最后变为系统的输出量 y(t)，从而通过 y(t) 可以观测所有的状态变量 X(t)。

与系统可控性的情况类似，除第一可观规范型外还有第二和其他的一些可观规范型，这里也就不再赘述了。同样，如果不特别指明的话，以下的系统可观规范型都是指系统的第一可观规范型。

[例 2.9] 对于上例的线性定常系统，试判断系统的可观性。如果系统状态完全可观，试求其可观规范型和变换矩阵。

结果：系统完全可观，其可观规范型和变换矩阵为：

$$A = \begin{pmatrix} 2 & 0 & 0 & 1 \\ 0 & 4 & 1 & 3 \\ 0 & 0 & 4 & 1 \\ 0 & 0 & 0 & 2 \end{pmatrix} \qquad B = \begin{pmatrix} 1 \\ 0 \\ 1 \\ 2 \end{pmatrix}$$

$$C = (1 \quad 1 \quad 0 \quad 0)$$

```
System is Observable.
System First Observable Canonical Form is:
Aol  = -0.0000      1.0000        0            0
        0            0            1.0000       0
        0.0000      -0.0000       0            1.0000
       -64.0000     96.0000      -52.0000     12.0000
Bol  = 1.0000
       11.0000
       58.0000
       268.0000
Col  =  1     0     0     0
The Transformation Matrix is:
```

```
P =  -14.0000    16.0000    -5.3750    0.5625
      15.0000   -16.0000     5.3750   -0.5625
      0           1.0000    -0.7500    0.1250
     -8.0000      8.0000    -2.5000    0.2500
```

求解过程:

在 MATLAB Command Window 下键入"edit"或选择"File"菜单新建 M - file,进入 MATLAB Editor/Debugger,编辑 M 文件"observeform.m":

```
%系统状态方程模型
A = [2 0 0 1;0 4 1 3;0 0 4 1;0 0 0 2];
B = [1;0;1;2];
C = [1 1 0 0];
%系统阶次
n = length(A);
%求解系统可观性矩阵
Q = zeros(n);
Q(1,:) = C;
for i = 2:n
%系统可观性矩阵的列向量
    Q(i,:) = Q(i-1,:) * A;
End
%系统可观性矩阵的秩
m = rank(Q);
%判断系统是否状态完全可观,并求解可观规范型
if m = = n
%系统完全可观
    P = inv(Q);
    Aol = inv(P) * A * P;
    Bol = inv(P) * B;
    Col = C * P;
%显示结果
    disp('System is Observable.');
    disp('System First Observable Canonnical Form is:');
    Aol
    Bol
    Col
    disp('The Transformation Matrix is:');
%状态变换矩阵
    P
else
%系统状态不完全可观
    disp('System State Variables cannot be totally Observed');
    disp('The rank of System Observable Matrix is:');
```

```
    %可观的状态变量数
        m
    end
```

选择"File"菜单的"Save"选项,保存文件名为"observeform.m"。选择"Tools"菜单的"Run"选项或在 MATLAB Command Window 下直接键入文件名 observeform,在 MATLAB Command Window 下查看运行的结果。

小结:对照例 2.8 的系统可控规范型可以看到,可观规范型矩阵 Aol 最后一行与可控规范型矩阵 Acl 最后一列的元素完全相同,都是[- 64.0000 96.0000 - 52.0000 12.0000]。事实上,在以后有关控制系统各数学描述间相互转换的章节中就会了解,这个行向量的元素就是该系统传递函数分母多项式的系数,也称为矩阵 A 的特征多项式的系数。

有关系统状态空间规范型的介绍就到此为止,研究规范型对系统的分析和综合都有十分重要的意义。规范型可以把系统的某些特性表现得更充分、更明显,规范型的各系数矩阵的元素往往有十分简洁的形式,给系统的分析和综合带来极大的方便。

采用状态空间法综合和设计系统时,常常采用非奇异线性变换将系统化为特定的规范型式。再对规范型进行综合和设计,这样的综合或设计方法是规范化的,便于使用计算机辅助计算。

从以上的几个实例中我们也可以看到,使用 MATLAB 求解系统的规范型是非常简洁并且有效的。有了 MATLAB 这个得力的工具,我们还可以根据不同的实际需要,自己定义系统规范型系数矩阵的特定结构,并构造适当的非奇异线性变换矩阵,将系统化为该规范型。用 MATLAB 探索和实现这些想法都是非常方便的。

2.4 控制系统模型转换

本章前三节主要介绍了控制系统的微分方程描述、传递函数描述和状态方程描述。其中微分方程描述是整个控制系统数学描述的基础,传递函数描述和状态方程描述都是在微分方程描述的基础上发展起来的。但由于形式不够简洁,处理和运算不够方便,一般在控制系统仿真和设计时已经很少直接用到微分方程描述了。但因为人们习惯使用的传递函数描述和状态方程描述分属于频率域和时间域,这样一来就存在一个模型间相互转换的问题。当然,正如前文所述,我们可以先把传递函数描述或状态方程描述化为微分方程描述,这只需将前几节的有关推导作逆运算即可,然后再化成希望的描述形式。不过 MATLAB 提供了直接进行控制系统模型间相互转换的函数,可以完成传递函数描述与状态方程描述之间的相互转换,控制系统方框图与传递函数描述和状态方程描述之间的转换,以及状态方程描述的最小实现。

2.4.1 传递函数向状态方程的转换

在实际的控制系统仿真和设计中,系统的结构和参数基本上是未知的。这时要想通过分析的方法建立系统的状态方程描述是非常困难的,甚至是不可能的。一个可行的办法是,先用实验的方法确定其输入与输出之间的关系,比如说确定它的传递函数。然后,

根据系统的传递函数再确定系统的状态方程和输出方程。由系统的传递函数或脉冲响应函数来建立与其输入输出特性上等价的状态方程描述称为实现问题。所找到的状态方程,称为该传递函数的一个实现。实现问题是控制理论和控制工程的一个基本问题。

给定系统的传递函数 G(s),可以找到各种各样结构的实现(A,B,C,D)。如果找到的实现对矩阵(A,B)是完全可控的,就称为可控性实现;如果对矩阵(A,C)是完全可观的,就称为可观性实现;如果矩阵 A 是约当规范型,就称为约当型实现。在各种各样的实现中,最感兴趣的是矩阵 A 的阶次最低的实现,称之为最小实现。显而易见,最小实现的结构最简单,按最小实现模拟原系统是最方便、最经济的。通过理论推导我们可以得到下边的结论:

(A,B,C,D)是传递函数 G(s)的最小实现的充分必要条件是,(A,B,C,D)是完全可控又可观的。

G(s)的最小实现也不是惟一的,它们维数相同,系数矩阵 A 之间是非奇异线性变换的关系。

MATLAB 提供了一条函数 tf2ss(),可以根据系统的传递函数描述求解其状态方程描述。其基本调用格式为:

[A,B,C,D] = TF2SS(NUM,DEN)

其中 NUM 是传递函数分子多项式系数的降幂排列,DEN 是传递函数分母多项式系数的降幂排列;(A,B,C,D)就是实现该传递函数的系统状态方程描述,这个实现是系统的可控性实现。

MATLAB 还提供了一条函数 minreal(),可以求解系统状态方程的最小实现。不过这条函数不是根据传递函数来求解其最小实现,而是根据系统的状态方程描述来求解其最小实现。其调用格式为:

SYSr = MINREAL(SYS)

其中 SYS 是原系统的状态方程描述(A,B,C,D),SYSr 是系统的最小实现(As,Bs,Cs,Ds)。请看下面这个简单的例子:

[例 2.10] 某控制系统的传递函数如下,试求其可控性实现。并判断该实现是否是系统的最小实现。如果不是,则给出系统的一个最小实现。

$$G(s) = \frac{1}{s^2(s^2 - s + 1)}\begin{pmatrix} s^2 - s \\ s^2 \end{pmatrix}$$

结果:系统的可控性实现为:

```
system Controller Realization is:
A =   -1    1    0    0
       1    0    0    0
       0    1    0    0
       0    0    1    0
B =    1
       0
       0
       0
C =    0    1   -1    0
```

```
     0    1    0    0
D =  0
     0
```

该实现不是系统的最小实现，最小实现为：

```
1 state(s) removed
system Minimal Realization is:
As = -1.0000     0.5257     0.8507
      0.5257     0.4472     0.7236
      0.8507    -0.2764    -0.4472
Bs = 1
     0
     0
Cs = 0    -0.3249    1.3764
     0     0.5257    0.8507
Ds = 0
     0
```

分析：本例给出的系统传递函数确切地说应该称作传递函数矩阵。当系统的输入量或输出量不只一个时(也就是 MIMO 系统)，就需要用传递函数矩阵来表示系统的输入量与输出量之间的关系。本例 2×1 的传递函数矩阵表示有两个输出量，一个输入量。从结果来看，无论是系统的可控性实现还是最小实现，与输出量相关联的矩阵 C 都是由 2 个行向量构成，而与输入量相关联的矩阵 B 都是一维的列向量。

本例要求判断系统的可控性实现是否是最小实现，可以通过判断该实现的可观性来解决。如果系统完全可观，那么该可控性实现就是系统的最小实现；否则就根据该可控性实现调用 minreal()函数求解系统的最小实现。

从结果来看，通过 tf2ss()函数求出的系统可控性实现的 A 矩阵和 C 矩阵的最后一列的元素均为零，这说明最后一个状态变量在状态方程中没有反映，无法通过输出量 Y(二维)来观测。minreal()函数也给出结果"1 state(s) removed"，消去了一个状态变量。也就是说，原系统四阶的传递函数通过三维的状态方程系数矩阵就可以实现了，系统的传递函数中存在零极相消。

求解过程：在 MATLAB Command Window 下键入"edit"或选择"File"菜单新建 M-file，进入 MATLAB Editor/Debugger，编辑 M 文件"realization.m"：

```
%系统的传递函数描述
num = [1 -1 0;1 0 0];
den = [1 1 -1 0 0];
%求解系统的可控性实现
[A,B,C,D] = tf2ss(num,den);
disp('system Controller Realization is:');
A
B
C
D
```

```
%求解可观性矩阵
n = length(A);
m = size(C,1);
%系统可观性矩阵
Q = zeros(m * n,n);
for i = 1:m
    Q(i,:) = C(i,:);
end
for i = 1:n - 1
    for j = 1:m
        Q(i * m + j,:) = Q((i - 1) * m + j,:) * A;
end
end
%判断是否是最小实现
r = rank(Q);
%是最小实现
if r = = n
    disp('Original (A,B,C,D) is system Minimal Realization');
%不是最小实现
else
    %求解系统的最小实现
    [As,Bs,Cs,Ds] = minreal(A,B,C,D);
    %显示结果
    disp('Original (A,B,C,D) is system Minimal Realization');
    disp('system Minimal Realization is:');
    As
    Bs
    Cs
    Ds
end
```

选择"File"菜单的"Save"选项,保存文件名为"realization.m"。选择"Tools"菜单的"Run"选项或在 MATLAB Command Window 下直接键入文件名 realization,在 MATLAB Command Window 下查看运行的结果。

小结:本例在判断系统的可观性时,使用的是改进了的例 2.9 的代码。例 2.9 的代码只能判断 SISO 系统的可观性,而本例的代码能够判断任意线性定常系统的可观性。读者也可以自己编制有关系统最小实现的 M 函数,这要用到马尔可夫(Markov)参数矩阵的一些知识,这里就不介绍了。

2.4.2 状态方程向传递函数的转换

给定线性定常系统(A,B,C,D),其状态方程和输出方程为:

$$\dot{X} = AX + Bu$$
$$Y = CX + Du$$

方程两边同时取拉普拉斯变换并代入消元，得

$$Y(s)=C(sI-A)^{-1}BU(s)+DU(s)$$

则系统的传递函数为：

$$G(s)=\frac{Y(s)}{U(s)}=C(sI-A)^{-1}B+D$$

其中$(sI-A)^{-1}$称为系统的预解矩阵。

可见，将系统的状态方程描述转换为传递函数描述的关键就在于预解矩阵的求解。我们在介绍矩阵指数时提到过，预解矩阵的拉普拉斯逆变换就是矩阵指数，而矩阵指数是求状态方程解的关键所在。因此，预解矩阵的分析和求解在系统的状态方程描述中具有非常重要的意义。

在手工计算时通常先按照法捷耶娃法来求解系统的预解矩阵，然后再根据状态方程系数矩阵求解系统的传递函数。MATLAB 提供了一条函数 ss2tf( )可以直接将系统的状态方程描述转换为传递函数描述，其基本调用格式为：

(NUM,DEN) = SS2TF(A,B,C,D,iu)

其中(A,B,C,D)是系统的状态方程描述的系数矩阵，iu 表示对系统的第 iu 个输入量求传递函数；NUM 是返回的系统传递函数的分子多项式系数的降幂排列，如果是 MIMO 系统 NUM 就是矩阵，其行数与输出量 y 的个数一样多；DEN 是返回的系统传递函数的分母多项式系数的降幂排列，无论是 SISO 系统还是 MIMO 系统，DEN 都是行向量。请看下面这个简单的例子：

[例 2.11]　给定某控制系统的状态方程描述如下。试分别求其对第一和第二个输入的传递函数和零极点形式的传递函数。

结果：系统的传递函数和零极点模型为：

$$A=\begin{pmatrix}0&0&0&1\\1&0&0&-2\\-22&-11&-4&0\\-23&-6&0&-6\end{pmatrix}\qquad B=\begin{pmatrix}0&0\\0&0\\0&1\\1&3\end{pmatrix}$$

$$C=\begin{pmatrix}0&0&0&1\\0&0&1&0\end{pmatrix}$$

```
System Transfer Function of the first input is:
num1  =   0          1.0000        4.0000        -0.0000       -0.0000
          0         -0.0000       -0.0000        -0.0000      -11.0000
den1  =      1.0000  10.0000       35.0000        50.0000       24.0000
Its zero-pole form is:
z1  =   0      Inf
        0      Inf
       -4      Inf
p1  =  -4.0000
       -1.0000
       -2.0000
       -3.0000
```

```
k1 =    1.0000
      -11.0000
System Transfer Function of the second input is:
num2 = 0      3.0000   12.0000   -0.0000   -0.0000
       0      1.0000    6.0000   11.0000  -27.0000
den2 = 1.0000   10.0000   35.0000   50.0000   24.0000
Its zero - pole form is:
z2 = -0.0000              -3.6557 + 2.6878i
      0.0000              -3.6557 - 2.6878i
     -4.0000               1.3114
p2 = -4.0000
     -1.0000
     -2.0000
     -3.0000
k2 = 3
     1
```

传递函数：

$$G(s)=\frac{1}{s^4+10s^3+35s^2+50s+24}\begin{pmatrix} s^3 & 3s^3+12s^2 \\ -11 & s^3+6s^2+11s-27 \end{pmatrix}$$

对应的零极点模型为：

$$\frac{1}{(s+4)(s+1)(s+2)(s+3)}\begin{pmatrix} s^2(s+4) & 3s^2(s+4) \\ -11 & (s+36.6-2.69i)(s+3.66+2.69i)(s-1.31) \end{pmatrix}$$

分析：可以看出，此系统有两个输入量，两个输出量。因此，系统的传递函数为 2×2 的矩阵。结果中系统对第一个输入量传递函数的零点有 inf，这表示该项是常数项，不包含 s 的幂。

求解过程：

在 MATLAB Command Window 下键入"edit"或选择"File"菜单新建 M－file，进入 MATLAB Editor/Debugger，编辑 M 文件"transfunc.m"：

```
%系统的状态方程描述
A = [0 0 0 1;1 0 0 -2;-22 -11 -4 0;-23 -6 0 -6];
B = [0 0;0 0;0 1;1 3];
C = [0 0 0 1;0 0 1 0];
D = [0 0;0 0];
%第一个输入量的传递函数
[num1,den1] = ss2tf(A,B,C,D,1);
[z1,p1,k1] = ss2zp(A,B,C,D,1);
%第二个输入量的传递函数
[num2,den2] = ss2tf(A,B,C,D,2);
[z2,p2,k2] = ss2zp(A,B,C,D,2);
%显示结果
disp('System Transfer Function of the first input is:');
```

```
num1
den1
disp('Its zero - pole form is:');
z1
p1
k1
disp('System Transfer Function of the second input is:');
um2
den2
disp('Its zero - pole form is:');
z2
p2
k2
```

选择"File"菜单的"Save"选项,保存文件名为"transfunc.m"。选择"Tools"菜单的"Run"选项或在 MATLAB Command Window 下直接键入文件名 transfunc,在 MATLAB Command Window 下查看运行的结果。

小结:本例在求解系统的零点、极点和增益时,没有使用前边介绍过的函数 tf2zp(),而是使用了一个新的函数 ss2zp()。该函数的功能是直接根据系统的状态方程描述求取系统的零点、极点和增益,不必先化成传递函数描述形式。其基本调用格式与 tf2zp()函数类似,只是多了一个输入量选择的参数:

[Z,P,K]= SS2TF(A,B,C,D,iu)

其中(A,B,C,D)是系统的状态方程描述的系数矩阵;iu 表示对系统的第 iu 个输入量求零点、极点和增益;Z 是求得的零点矩阵;P 是求得的极点矩阵,Z 和 P 的列数与输出量 y 的个数一样多,行数视零点个数的多少而定;K 是求得的增益列向量,其维数等于输出量 y 的个数。

2.4.3 由方框图求状态方程和传递函数

在研究复杂的动态系统时,一般习惯利用框图来辅助列写系统的微分方程或传递函数。这样做的好处:一方面是直观,符合控制系统的实际情况;另一方面也方便系统模型的化简和运算,包括系统的综合和校正等等。系统方框图最基本的单元如图 2-18 所示,其闭环传递函数为:

$$H(s) = \frac{1}{1+(s)}$$

复杂系统的框图可能有许多个框,框与框之间的连接关系也错综复杂。这就给手工计算系统的传递函数和状态方程带来了很大的困难。考虑到计算的烦琐性,而数字式计算机在这方面的优势非常明显,因此,这些工作目前一般都由计算机辅助设计软件来完成。MATLAB 提供了一条函数 connect()可以将方框图描述的线性系统转换为状态方程的描述形式。其基本调用格式为:

SYSc = CONNECT(SYS,Q,INPUTS,OUTPUTS)

相关参数和返回值的说明请看下面例子的小结。

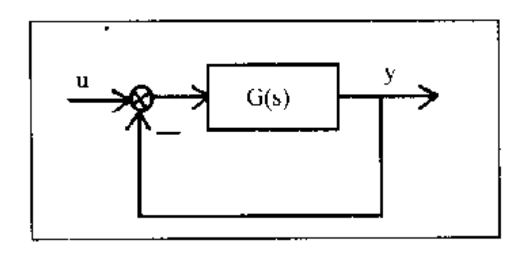


图 2-18 控制系统方框图的基本单元

[例 2.12] 某控制系统的方框图如图 2-19 所示。试求解其状态方程描述(A,B,C,D)和系统的传递函数。

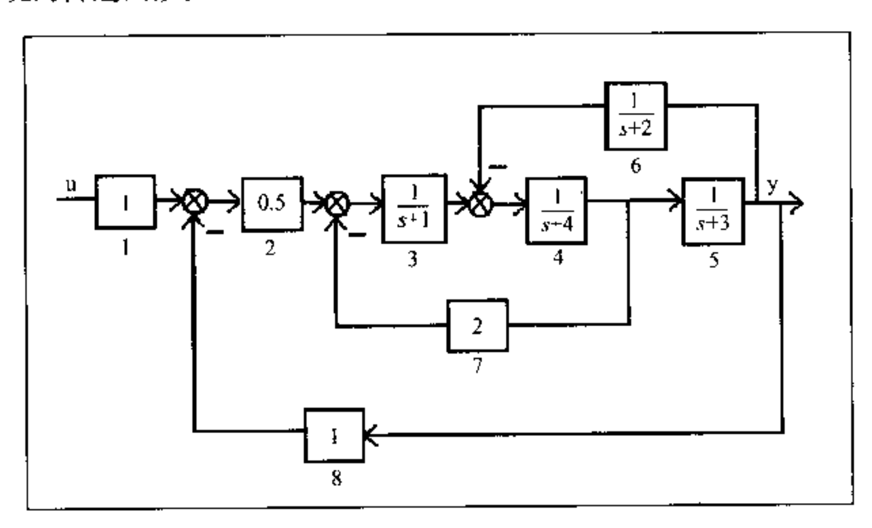


图 2-19 线性系统方框图

结果:系统的状态方程描述为:

```
State model [a,b,c,d] of the block diagram has 8 inputs and 8 outputs
State - Space Model of the block - diagram is:
A = -1.0000    -1.0000    -0.5000         0
     1.0000    -4.0000     0         -1.0000
     0          1.0000    -3.0000         0
     0          0          1.0000    -2.0000
B = 0.5000
    0
    0
    0
C = 0     0     1     0
D = 0
```

系统的传递函数描述为:

```
Transfer Function of the block - diagram is:
num =   0        0.0000    0.0000    0.5000    1.0000
den =   1.0000  10.0000   36.0000   56.5000   32.0000
```

对应传递函数为：　$G(s)=\dfrac{0.5s+1}{s^4+10s^3+36s^2+56.5s+32}$

分析：对于复杂系统框图仿真的问题，首先要将原系统方框图分解为单个的基本框图，并求解其关联矩阵；然后再寻找合适的 MATLAB 函数进行仿真。

求解过程：

本例的解题过程分为两部分：

1. 求解系统的关联矩阵

系统的关联矩阵 Q 是描述各框图间连接关系的矩阵。其每行第一个元素是框号，其余的元素依次是与该框连接的框号。图 2－19 各框的编号已经给出，例如第 3 框，它与第 2 框正连接，与第 7 框负连接，与其他框不相连，那么关联矩阵 Q 的第三行就是[3 2 －7]。依次类推，得到系统的关联矩阵为：

$$Q=\begin{pmatrix}1 & 0 & 0\\ 2 & 1 & -8\\ 3 & 2 & -7\\ 3 & 2 & -6\\ 5 & 4 & 0\\ 6 & 5 & 0\\ 7 & 4 & 0\\ 8 & 5 & 0\end{pmatrix}$$

2. 求解系统的状态方程和传递函数

在 MATLAB Command Window 下键入“edit”或选择“File”菜单新建 M－file，进入 MATLAB Editor/Debugger，编辑 M 文件“diag2ss.m”：

```
%清除所有内存变量
clear all;
%方框图初始化
n1 = 1;d1 = 1;
n2 = 0.5;d2 = 1;
n3 = 1;d3 = [1 1];
n4 = 1;d4 = [1 4];
n5 = 1;d5 = [1 3];
n6 = 1;d6 = [1 2];
n7 = 2;d7 = 2;
n8 = 1;d8 = 1;
%方框个数
nblocks = 8;
%建立方框图
blkbuild;
%建立方框图的对角非连接形式的状态空间模型
sys = ss(a,b,c,d);
%建立方框图连接矩阵
q=[1 0 0;2 1 -8;3 2 -7;4 3 -6;5 4 0;6 5 0;7 4 0;8 5 0];
```

```
%求解方框图的状态空间模型
sysc = connect(sys,q,1,8);
%数据扩展
[A,B,C,D] = ssdata(sysc);
%将状态空间模型转换为传递函数模型
[num,den] = ss2tf(A,B,C,D,1);
%显示结果
disp('State - Space Model of the block - diagram is:');
A
B
C
D
disp('Transfer Function of the block - diagram is:');
num
den
```

选择"File"菜单的"Save"选项,保存文件名为"diag2ss.m"。选择"Tools"菜单的"Run"选项或在 MATLAB Command Window 下直接键入文件名 diag2ss,在 MATLAB Command Window 下查看运行的结果。

小结:在 connect()函数的基本调用格式 SYSc = CONNECT(SYS,Q,INPUTS,OUTPUTS)中,SYS 是系统个框图无连接时的状态空间描述,Q 是系统的关联矩阵,INPUTS 和 OUTPUTS 分别是系统输入量和输出量所在方框的编号;SYSc 是返回的系统的状态方程描述。一般说来,在调用 connect()函数之前,首先要调用 blkbuid 语句使用,其功能是建立系统的方框图描述。blkbuid 语句是脚本(script)函数,其输入参数和返回值都不在函数体内说明。其输入参数有 nblocks,代表系统方框图的块数;(ni,di)或(ai,bi,ci,di),代表单个方框的传递函数描述或状态方程描述。其返回值为(a,b,c,d),代表将这些方框对角连接后的系统状态方程描述(注意,并不是按原连接方式求出的系统状态方程描述,因为此时并未调用系统的关联矩阵)。

调用 blkbuild 语句时需要注意的是,因为输入参数在函数体外声明,所以应该在程序的最开始清除所有内存变量。否则即使在程序中改变了输入参数,也可能得到与未改变前相同的结果。事实上,调用 blkbuild 语句与使用 append()函数的结果是一样的。append()函数将系统的方框图转化为如下对角连接的形式:

$$A=\begin{pmatrix}A_1 & & & \\ & A_2 & & \\ & & \ddots & \\ & & & A_n\end{pmatrix}\qquad B=\begin{pmatrix}B_1 & & & \\ & B_2 & & \\ & & \ddots & \\ & & & B_n\end{pmatrix}$$

$$C=\begin{pmatrix}C_1 & & & \\ & C_2 & & \\ & & \ddots & \\ & & & C_n\end{pmatrix}\qquad D=\begin{pmatrix}D_1 & & & \\ & D_2 & & \\ & & \ddots & \\ & & & D_n\end{pmatrix}$$

然后再调用 connect()函数,根据关联矩阵 Q 的信息求出系统的状态方程描述。程序

中还用到了 ssdata()函数,其功能是将系统的抽象描述 SYS 扩展成状态方程描述的系数矩阵。事实上,直接使用系统的抽象描述也可以得到希望的结果,例如执行完 diag2ss.m 文件后,在 MATLAB Command Window 下键入"sysc",有:

```
sysc
a =            x1      x2      x3      x4
        x1     -1      -1      -0.5    0
        x2     1       -4      0       -1
        x3     0       1       -3      0
        x4     0       0       1       -2
b =            u1
        x1     0.5
        x2     0
        x3     0
        x4     0
c =            x1      x2      x3      x4
        y1     0       0       1       0
d =            u1
        y1     0
Continuous - time system.
```

此(a,b,c,d)与结果中 ssdata()扩展出的(A,B,C,D)是完全一致的。但需要注意的是,并不能直接调用系统的参数矩阵(a,b,c,d),这 4 个矩阵也不会覆盖程序中的同名变量。例如在 MATLAB Command Window 下键入"a",有:

```
a
a =   -1     0     0     0
       0    -4     0     0
       0     0    -3     0
       0     0     0    -2
```

这是 blkbuild 语句的返回值(a,b,c,d)中的 a 矩阵。如果希望使用系统的参数矩阵,还是要先用 ssdata()语句进行数据扩展。

2.5　控制系统的稳定性

前面已经讲过如何在数学上描述控制系统并解出它的运动。只要知道了系统的结构和各参数,我们就能算出它的各物理量的变化规律。

从工程角度看,这还是不够的。一方面,系统越复杂,微分方程阶次就越高,求解也就越困难。对于实际工程领域的许多复杂的系统,微分方程阶次高达十几阶甚至几十阶,不用计算机实际上就是无法求解的。另一方面,实际工程问题并不是简单地求解一个既定系统的运动,而往往是要选择系统中的某些参数,甚至还要改变系统的结构,以求获得较好的静态和动态性能。

对于上述这些问题,如果都靠直接求解微分方程来研究,势必要解大量的微分方程,从而大大增加计算量。同时,只从微分方程也不容易区分影响系统运动规律的主要因素

和次要因素。

因此，就需要研究一些比较方便的工程分析方法。这些工程分析方法的计算量应当不太大，并且不因方程阶次的升高而增加太多。用这些方法不仅比较容易分析各主要参数对系统运动规律的影响，而且还可以借助一些图表和曲线直观地把运动特征表示出来。这些都是直接求解微分方程所做不到的。

在 MATLAB 环境下，虽然求解微分方程是非常容易的，但掌握有关系统稳定性的概念对于理论分析和工程实际应用来说都是非常必要而且是十分重要的。

关于运动稳定性严格定义有不只一种，但其中最重要的还是俄国学者 A.M.李亚普诺夫于 1892 年提出的经典定义：

如果一个关于 x 的微分方程组，在初始条件 $x(t_0)=x_0$ 下有解 $x(t)$，且对于任意给定的正数 $\epsilon>0$，总存在一个正数 $\delta(\epsilon)$，当初始条件 x_0 变为 $\bar{x}_0$ 时，只要 $\|\bar{x}(t)-x(t)\|\leqslant\delta$，其相应解 $\bar{x}(t)$在 $t>t_0$ 的任何时刻都满足 $\|\bar{x}(t)-x(t)\|<\epsilon$，则称解 $x(t)$是稳定的。如果不存在这样的正数 $\delta(\epsilon)$，则称解 $x(t)$是不稳定的。

对于线性控制系统来说，其稳定性的充分必要条件是：它的微分方程的特征方程的全部根都是负实数或实部为负的复数，亦即：全部根都位于复数平面的左半平面。

如果特征方程在复数平面的右半平面上没有根，但在虚轴上有根，则可以说该线性系统是临界稳定的。

例如对于二阶系统：
$$y=1-\frac{1}{\sqrt{1-\zeta^2}}e^{\zeta\omega t}\sin(\omega_d\omega t+\theta)$$

其中：
$$\theta=\mathrm{arctg}\frac{\sqrt{1-\zeta^2}}{\zeta},\omega_d=\sqrt{1-\zeta^2}$$

当 $0<\zeta$ 时系统是稳定的；$0<\zeta<1$ 时系统的响应曲线振荡衰减；$\zeta>1$ 时系统响应曲线阻尼衰减；当 $\zeta=0$ 时系统临界稳定，系统响应曲线等幅振荡；$\zeta<-1$ 系统不稳定，系统的响应曲线是发散的，如图 2-20 所示。

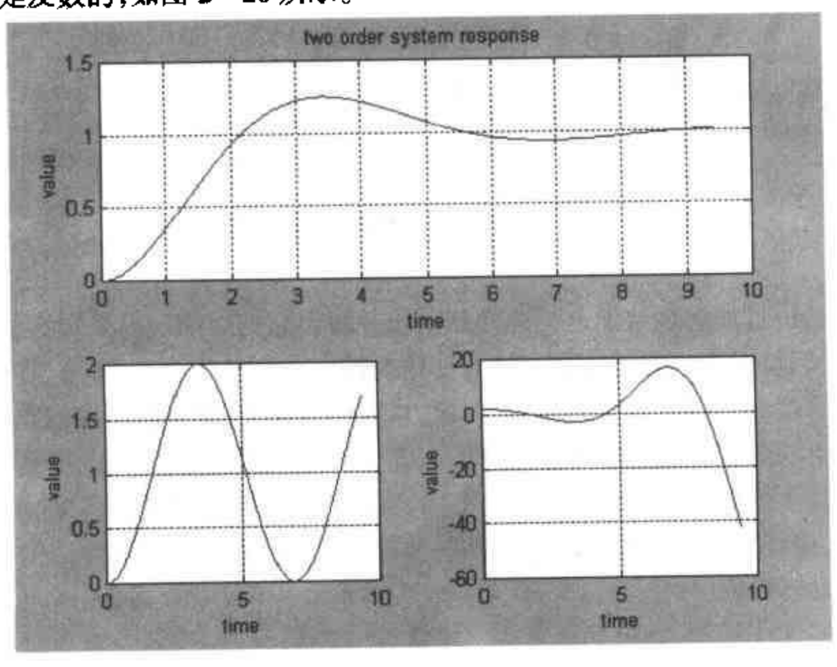


图 2-20 二阶系统稳定性分析

在计算机辅助设计软件出现之前，通常采用间接的方法判断系统的稳定性。例如 Routh 判据等等。在 MATLAB 环境下，我们不必再采用这样的方法，可以直接求解系统特征方程的根，根据根的分布来判断系统的稳定性。前边介绍过的 tf2zp()函数、ss2zp()函数等等都可以完成这项工作。如果仅仅知道系统的特征方程，还可以调用 roots()函数来求解特征方程的根，然后再判断系统的稳定性。知道了系统的零点之后，还可以判断该系统是否为最小相位系统。所谓最小相位系统，就是系统的零点均在复平面的左半平面，当然，系统首先是稳定的。请看下例：

[例 2.13]　某控制系统的状态方程描述如下。试判断其稳定性和是否为最小相位系统，并绘制其时间响应曲线来验证上述判断。

$$A=\begin{pmatrix}-3 & -8 & -2 & -4\\ 1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 1 & 0\end{pmatrix}\qquad B=\begin{pmatrix}1\\0\\0\\0\end{pmatrix}$$

$$C=(0\quad 0\quad 1\quad 1)$$

结果：系统稳定并且是最小相位的。

```
System zero - points,pole - points and gain are:
z =  -1
p =  -1.4737 + 2.2638i
     -1.4737 - 2.2638i
     -0.0263 + 0.7399i
     -0.0263 - 0.7399i
k = 1
System is stable
System is minimal Phase
```

系统的时间响应曲线如图 2－21 所示。

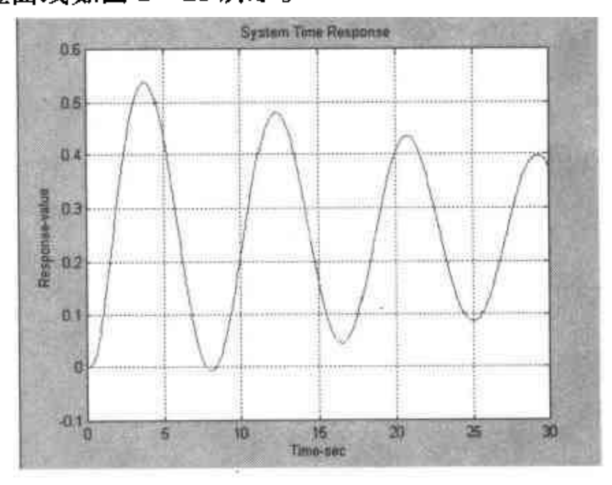


图 2－21　线性系统时间响应曲线

分析：系统的极点均在复平面的左半平面，因而系统是稳定的。从系统的时间响应曲线上看也是如此。系统的输出振荡衰减。但可以看出，系统衰减响应曲线的振荡频率很高，衰减速率很慢。这是因为系统有两个极点的实部为－0.0263，非常靠近虚轴的缘故。

在过程上,这种系统属于性能非常差的一类。稳定裕量很小,稍有干扰系统的极点就可能越过虚轴跑到复平面的右半平面去。从而造成系统的不稳定。对于这类系统,一般都要加校正手段,使其极点远离虚轴。

求解过程:

在 MATLAB Command Window 下键入"edit"或选择"File"菜单新建 M - file,进入 MATLAB Editor/Debugger,编辑 M 文件"teststable.m":

```
%系统状态方程模型
A = [ -3  -8  -2  -4;1 0 0 0;0 1 0 0;0 0 1 0];
B = [1;0;0;0];
C = [0 0 1 1];
D = [0];
%标志变量——判断是否稳定
flag1 = 0;
%标志变量——判断是否最小相位
flag2 = 0;
%求解零极点和增益
[z,p,k] = ss2zp(A,B,C,D,1);
%显示结果
disp('System zero - points, pole - points and gain are:');
z
p
k
%判断是否稳定
n = length(A);
for i = 1:n
    if real(p(i)) > 0
        flag1 = 1;
    end
end
%判断是否最小相位
m = length(z);
for i = 1:m
    if real(z(i)) > 0
        flag2 = 1;
    end
end
%显示结果
if flag1 = = 1
    disp('System is unstable');
else
    disp('System is stable');
end
```

```
if flag2 = = 1
    disp('System is Nonminimal Phase');
else
    disp('System is minimal Phase');
end
%系统仿真变量初始化
x0 = [0 0 0 0];
t = 0:0.1:30;
r = length(t);
u = ones(1,r);
[y,T] = lsim(A,B,C,D,u,t,x0);
%图形绘制
plot(t,y);
title('System Time Response');
xlabel('Time - sec');
ylabel('Response - value');
grid;
```

选择“File”菜单的“Save”选项，保存文件名为“teststable.m”。选择“Tools”菜单的“Run”选项或在 MATLAB Command Window 下直接键入文件名 teststable，在 MATLAB Command Window 下查看运行的结果。

小结：本例使用了 lsim()函数进行线性系统仿真，也可以用求解系统状态方程的方法进行系统仿真，效果是一样的。本例还使用了 ones()函数，其功能是产生任意维数的元素值均为 1 的矩阵。

判断非线性系统的稳定性时经常要用到李亚普诺夫稳定性基本定理，但该定理的陈述不是构造性的，需要使用者自己去寻找满足条件的李亚普诺夫函数。这一过程是相当困难的，在这方面目前的控制系统计算机辅助设计软件也没有提供什么得力的工具。不过在线性定常系统方面，MATLAB 提供了一个求解李亚普诺夫方程（也称塞尔维斯特(Sylvester)方程）的函数 lyap()，可以用来求解形如：

$$A^{T}P + PA = -Q$$

的方程。其中 Q 是任意正定矩阵，如果能找到正定矩阵 P 满足上述方程，则该线性定常系统的平衡态是渐进稳定的。例如系统状态方程矩阵为：

$$A = \begin{pmatrix} 0 & 1 \\ -1 & -1 \end{pmatrix}$$

则在 MATLAB Command Window 下键入：

```
A = [0 1; -1 -1];
Q = [1 0;0 1];
P = lyap(A,Q)
```

求得的正定矩阵 P 为：

```
P =    1.5000        -0.5000
      -0.5000         1.0000
```

P 有解，说明该系统的平衡态是渐进稳定的。这里为了方便选 Q 矩阵为单位阵，当

然也可以选别的正定矩阵。

2.6 小　　结

有关控制系统的数学描述就讨论到这里，本章我们主要讨论了 MATLAB 环境下控制系统运动方程、控制系统的传递函数描述、控制系统的状态方程描述以及各个控制系统模型之间的转换问题，并在此基础上，给出了控制系统稳定性的判据和数值检验方法。

其中控制系统运动方程即微分方程描述是控制系统数学描述的基础，其他各种描述形式都是在此基础上推演或发展起来的。虽然目前在工业领域使用较多的是控制系统的传递函数描述和状态方程描述这两种手段，但对于一个完全陌生的控制系统来说，还是要首先分析其机理，列出运动方程，然后才能化为传递函数描述或状态方程描述。从下一章开始将介绍有关 MATLAB 环境下控制系统的时频响应分析。

第三章　控制系统时频分析及根轨迹的绘制

上一章我们已经讲了描述动态系统运动的两类方法。其中一类是用微分方程描述，状态方程也包括在内；另一类是用传递函数描述，传递函数矩阵是它的一种推广，框图是它的一种图解。

有些情况下也常常用典型响应描述一个动态系统的性质。所谓响应，就是指由于输入量的作用而造成的对象输出量的变化的函数。典型响应是指零初值条件下某种典型的输入量函数作用下对象的响应。

当我们借助于传递函数的概念建立了：

$$y(s) = G(s)u(s)$$

这样的关系式以后，就拉普拉斯变换的像函数而言，输出量 y(s)与输入量 u(s)之间好像有了一种"比例"关系，其"比例系数"就是传递函数 C(s)。这样就很容易想到：能否以某种"单位"输入量信号下的输出量函数表示传递函数，从而以一种直观的方式表达系统的特性。这就是典型响应的概念。

常用的典型形象有脉冲响应、阶跃响应和频率响应。在这一章的时频分析中我们将详细讨论这三种响应。

我们知道，一个控制系统的全部性质，都取决于其闭环传递函数：稳定性取决于其极点；静态精度取决于其增益；动态性能即取决于其极点，也与其零点有关。我们又知闭环传递函数的零点与开环传递函数的零点相同，增益之间也有简单的关系，都不难确定。惟有闭环传递函数的极点，即闭环特征方程的根，手工计算起来比较困难。根轨迹法是 W. R. Evans 于 1948 年提出的一种求解闭环特征方程根的简便的图解方法，在工程上获得了广泛的应用。他根据系统开环传递函数极点和零点的分布，依照一些简单的规则，用作图的方法求出闭环极点的分布，从而避免了复杂的数学计算。系统中某个参数的变化对闭环极点的分布会产生什么影响，可以很容易地从图上看出来。虽然在 MATLAB 环境下可以很方便的求出闭环传递函数特征方程的根，但掌握根轨迹的规则和绘制方法对于控制系统的设计和校正还是十分必要的。

3.1　时域响应分析

在上一章里，我们曾经介绍了 MATLAB 环境下微分方程的描述和解法。事实上，所谓控制系统的时域响应分析就是在时间域内求解系统的微分方程，然后根据绘制出来的曲线分析系统的性能和各主要参数对系统性能的影响。不过这里的响应曲线一般是指典型的响应曲线，即所谓阶跃响应和脉冲响应。其激励函数分别为单位阶跃函数 u(t)（或 1(t)）和单位脉冲函数 $\delta(t)$。单位阶跃函数的数学表达式为：

$$u(t)=\begin{cases}0 & (t<0)\\ 1 & (t\geq 0)\end{cases}$$

对应拉普拉斯变换为 1/s。单位脉冲响应的数学表达式为：

$$\delta(t)=\begin{cases}0 & (t\neq 0)\\ \infty & (t=0)\\ \int_{-\infty}^{\infty}\delta(t)\,dt=1\end{cases}$$

对应的拉普拉斯变换为 1。

在上一章里求解系统典型响应的思路是首先列出微分方程，对方程两边同时取拉普拉斯变换得到系统的传递函数，然后根据输入量的类型确定输出量的表达式，求解此表达式并编制绘图程序，最后根据绘制出来的图形对系统进行分析。事实上，MATLAB 提供了一些函数可以直接求解系统的典型响应，可以让使用者从上述复杂的程序编制过程中解放出来，把更多的精力放在系统的性能分析上。

[例 3.1] 例如 MATLAB 控制系统工具箱提供了一条求解系统单位阶跃响应的函数 step()，其基本调用格式有：

STEP(SYS1,SYS2,...,T) 或 [Y,T,X] = STEP(SYS,...)

两种。其中第一种格式是不关心系统阶跃响应的具体数值，仅仅在一张图上绘制从系统 SYS1 到系统 SYSn 的阶跃响应曲线，T 是 t0:tspan:tfinal 格式的时间向量。第二种格式返回系统的阶跃响应数据，但并不在屏幕上绘制系统的阶跃响应曲线。其中 T 是返回的仿真时间向量，设其维数为 LT，并假设系统有 Nu 维输入量，Ny 维输出量，则返回值 Y 是 LT×Nu×Ny 维矩阵，其 Y(:,:,j)行向量表示第 j 个输入通道的时间响应数据；对于含有 Nx 个状态变量的系统的状态方程描述，X 是 LT×Nx×Ny 维矩阵，对应各状态变量的阶跃响应数据。在这两种格式中，SYS 可以是系统的传递函数描述[NUM,DEN]，NUM 和 DEN 分别代表传递函数的分子和分母多项式系数的降幂排列；也可以是系统的状态方程描述(A,B,C,D)；还可以是系统的抽象描述。

对于系统的单位脉冲响应，MATLAB 也提供了一条函数 impulse()实现此功能。其基本调用格式也是 IMPULSE(SYS1,SYS2,...,T) 或 [Y,T,X] = IMPULSE(SYS,...)两种，具体参数和返回值的含义与 step()函数相同。请看下例：

系统的传递函数如下。试求其闭环传递函数，并绘制输出量阶跃响应曲线和脉冲响应曲线。选择函数的状态变量将其化为状态方程模型，并绘制状态变量的阶跃响应曲线和脉冲响应曲线。

$$G_a(s)=\frac{200}{s^4+20s^3+140s^2+400s+384}$$

结果：系统的闭环传递函数系数为：

```
System Closed Loop Transfer Function is:
numc =   0      0      0      0      200
denc =   1      20     140    400    584
```

$$G_a(s)=\frac{200}{s^4+20s^3+140s^2+400s+384}$$

系统的状态方程模型为：

```
System State - Space Model is:
A = -20     -140     -400     -584
     1         0        0        0
     0         1        0        0
     0         0        1        0
 B =  1
      0
      0
      0
 C =  0    0    0   200
 D =  0
```

系统输出量的阶跃响应如图 3-1 所示。其横坐标为时间,纵坐标为系统的输出量阶跃响应值。

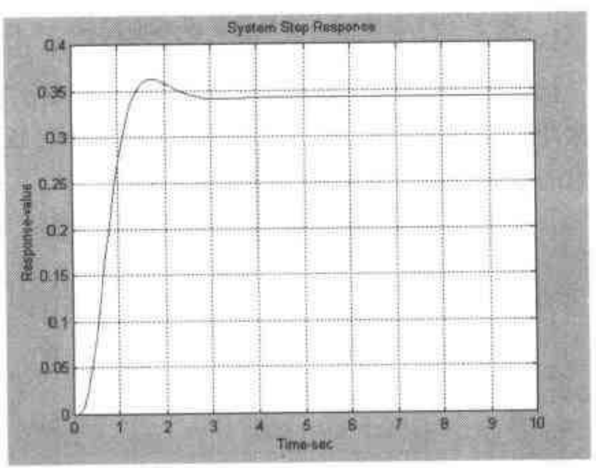


图 3-1 系统输出量的阶跃响应

系统输出量的脉冲响应如图 3-2 所示。其横坐标为时间,纵坐标为系统的输出量脉冲响应值。

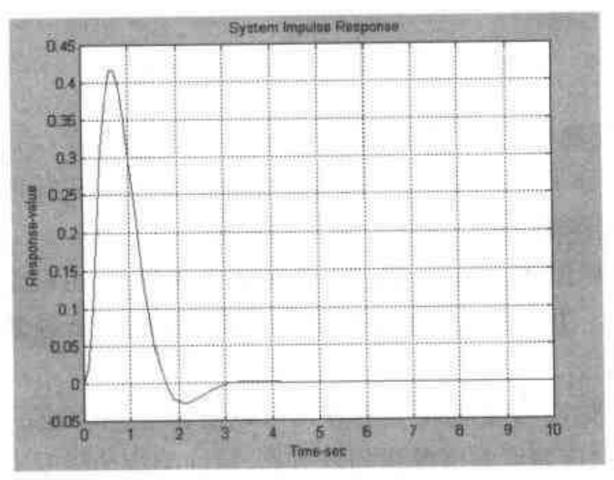


图 3-2 系统输出量的脉冲响应曲线

系统状态变量的脉冲响应曲线如图 3-3 所示。其横坐标为时间,纵坐标为系统的状态变量阶跃响应值。

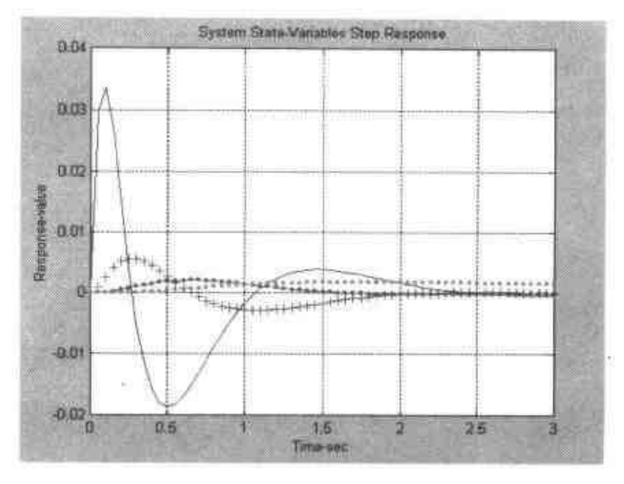


图 3-3 系统状态变量的阶跃响应曲线

因为本系统有四个状态变量,所以用四种不同描点格式来显示它们的响应值。

系统状态变量的脉冲响应曲线见图 3-4。其横坐标为时间,纵坐标为系统的状态变量脉冲响应值。同样用四种不同描点格式来显示它们的响应值。

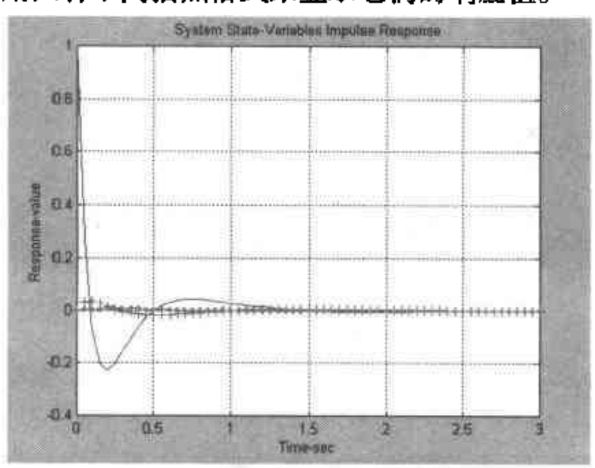


图 3-4 系统状态变量的脉冲响应曲线

分析:在单位负反馈情况下,系统的闭环传递函数表达式为 $G(s)/(1+G(s))$。因此,系统闭环传递函数的分子多项式与开环传递函数的分子多项式相同;闭环传递函数的分母多项式是开环传递函数的分母多项式加分子多项式。

从系统输出量的阶跃响应来看,响应曲线振荡幅度很小,也就是超调量比较小,大约 $0.02/0.34=6\%$左右。并且达到峰情后迅速下调,在 3s 附近已经基本达到稳态值,也就是过渡过程时间不超过 3s。

从系统输出量的脉冲响应曲线上来看也可以验证这些说法:脉冲响应曲线大约在 1.7s 左右过零,此时对应阶跃响应曲线的峰值点;脉冲响应曲线只有一个过零点,对应阶跃响应曲线只振荡一次就达到稳态值;并且脉冲响应曲线也是在大约 3s 左右达到稳态值 0。

因为系统闭环传递函数的特征方程是四阶的，所以化成状态方程模型后有四个状态变量。对应系统状态变量的阶跃响应和脉冲响应就各有 4 条曲线。其中实线对应第一个状态变量，“+”对应第二个状态变量，“.-”对应第三个状态变量，“.”对应第四个状态变量。这些状态变量的阶跃响应曲线和脉冲响应曲线都是经过短暂振荡后迅速回到零点，这从另一个侧面验证了上面的那些说法。其中第一个状态变量的振荡的峰值最大，次数也最多。这是因为 tf2ss()函数将系统的传递函数模型转化为状态空间的可控规范型，第一个状态变量与系统输入量 u(t)最接近。零时刻输入量加入到系统中，引起状态变量的突然变化。对于其他的状态变量来说，它们和输入量 u(t)之间隔着 1～3 阶的惯性环节，阶跃函数和脉冲函数的冲激效应就缓和许多了。因此振荡的峰值就比较小。

一般说来，具有上述这些性质的系统是我们在控制系统中希望看到的，即系统输出的超调量很小，过渡过程时间很短，能够迅速跟踪输入量设定值的变化。当然，这一切都是在系统稳定的前提下而言的。另外，本例系统的阶次比较高，容易引进高频干扰。实际应用中一般用二阶模型近似后再进行处理。

求解过程：

在 MATLAB Command Window 下键入“edit”或选择“File”菜单新建 M－file，进入 MATLAB Editor/Debugger，编辑 M 文件“sysresponse.m”：

```
%关闭所有图形窗口并清除内存变量
close all;
clear;
%系统开环传递函数初始化
numo=[0 0 0 0 200];
deno=[1 20 140 400 384];
%求解系统的闭环传递函数
numc=numo;
n=length(deno);
denc=zeros(1,n);
denc=numo+deno;
%结果显示
disp('System Closed Loop Transfer Function is:')
numc
denc
%系统仿真数据初始化
t=0:0.05:3;
%系统输出量的阶跃响应
y=step(numc,denc,t);
%系统输出量的脉冲响应
yy=impulse(numc,denc,t);
%输出量阶跃响应曲线绘制
plot(t,y);
title('System Step Response');
xlabel('Time-sec');
```

```
ylabel('Response - value');
grid;
%输出量脉冲响应曲线绘制
plot(t,yy);
title('System Impulse Response');
xlabel('Time - sec');
ylabel('Response - value');
grid;
%求解系统的状态方程模型
[A,B,C,D] = tf2ss(nume,denc);
%状态方程模型结果显示
disp('System State - Space Model is:');
A
B
C
D
%系统状态变量的阶跃响应
[ys,x] = step(A,B,C,D,1,t);
%系统状态变量的脉冲响应
[yys,xx] = impulse(A,B,C,D,1,t);
%状态变量节约响应曲线绘制
plot(t,x(:,1),t,x(:,2),'+',t,x(:,3),'-',t,x(:,4),'.')
title('System State - Variables Step Response');
xlabel('Time - sec');
ylabel('Response - value');
grid;
%状态变量脉冲响应曲线绘制
plot(t,xx(:,1),t,xx(:,2),'+',t,xx(:,3),'-',t,xx(:,4),'.')
title('System State - Variables Impulse Response');
xlabel('Time  sec');
ylabel('Response - value');
grid;
```

选择"File"菜单的"Save"选项,保存文件名为"sysresponse.m"。选择"Tools"菜单的"Run"选项或在 MATLAB Command Window 下直接键入文件名 sysresponse,在 MATLAB Command Window 下查看运行的结果。

小结:本例分别调用了两种不同格式的 step()函数和 impulse()函数,并且根据不同的系统描述模型绘制了输出量和状态变量的阶跃响应和脉冲响应。一般说来,在控制系统仿真和设计的时间域分析有这两种响应曲线就足够了,系统的主要参数及其变化规律在这两种响应曲线中都有所反映。

从工程的角度来看,实际控制系统的输入函数千变万化,实际模拟控制系统时仅靠上述这两种曲线虽然也是一种解决问题的思路(从理论上说,阶跃函数和脉冲函数之和可以以任意精度逼近某一函数),不过计算的复杂性就大大增加了。因此,MATLAB 还提供了

一个函数可以求解和绘制任意输入函数激励的系统时间响应，这就是前文介绍过的 lsim()函数。

［例 3.2］　例如希望求解传递函数为：

$$G(s)=\frac{s+1}{s^3+7s^2+14s+8}$$

的控制系统在输入函数 u = sint 时的时间响应，可以在 MATLAB Command Window 下键入下列语句：

```
num = [0 0 1 1];
den = [1 7 14 8];
t = 0:0.05:5;
u = sin(t);
lsim(num,den,u,t)
```

系统的响应曲线，如图 3－5 所示。

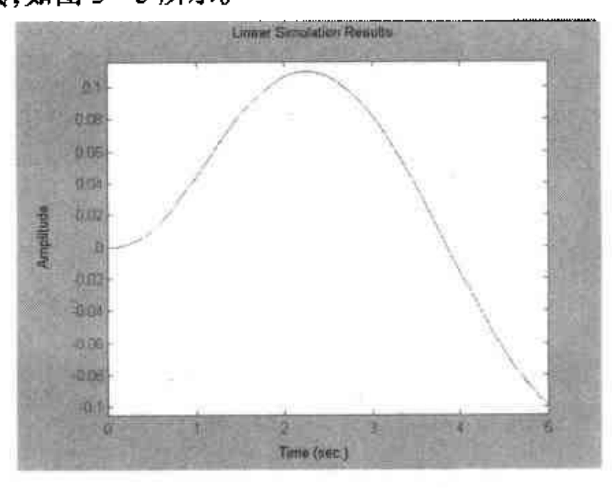


图 3－5　正弦函数激励下三阶线性系统的响应

其横坐标为时间，纵坐标为系统在单位正弦函数 u = sint 激励下的响应值。从图中可以看出，大约在 2.3s 左右系统响应曲线达到最大值，这也是系统在单位正弦激励下的共振点。本书第四章对系统进行频率分析时将具体讲述这方面内容。

当然，在 MATLAB 环境下也可以调用[Y,T] = LSIM(SYS,U,...)格式的 lsim()函数，根据返回的时间向量 T 和任意函数响应值 Y，通过 plot()语句绘制任意函数激励的响应曲线。这样的好处是可以随意控制曲线的绘制格式。

对于离散时间的控制系统（或称采样控制系统），MATLAB 同样也提供了相应的 dstep()、dimpulse()和 dlsim()函数求解其单位阶跃响应、单位脉冲响应和任意函数的激励响应。

这三条函数的参数设置、返回值和调用格式与连续控制系统相应的 step()、impulse()和 lsim()函数完全一致，只不过将相应的传递函数描述和状态方程描述换为脉冲传递函数描述和离散状态方程描述，并且求得的响应数据也是离数的。例如希望求解离散控制系统：

$$\bar{x}(z)=\frac{1.6z^2-z}{z^2-0.8z+0.5}$$

的单位阶跃响应和单位脉冲响应,可以在 MATLAB Command Window 下键入下列语句:

```
num=[1.6 -1 0];
den=[1 -0.8 0.5];
subplot(211),
dstep(num,den);
subplot(212),
dimpulse(num,den);
```

此离散系统的响应曲线如图 3-6 所示。

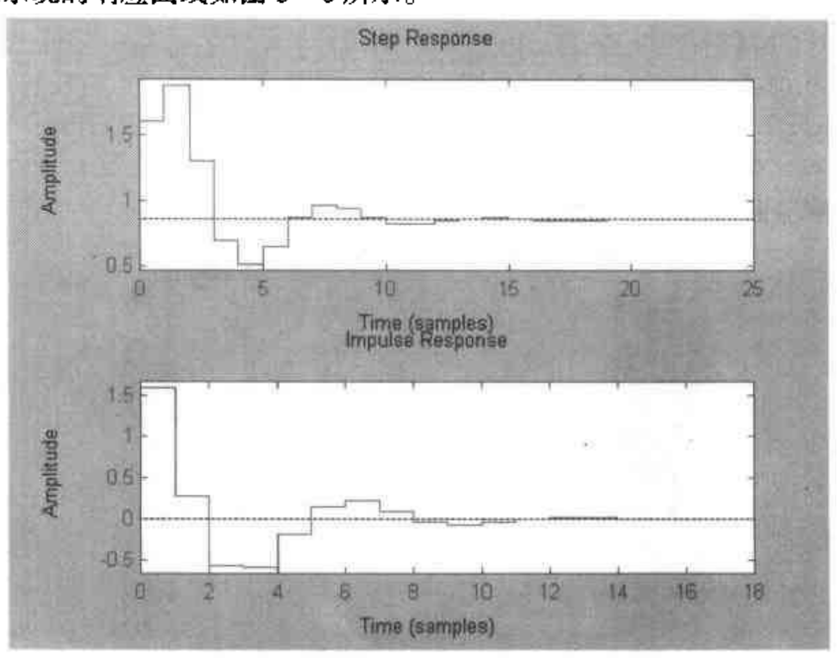


图 3-6 离散控制系统的阶跃响应和脉冲响应

其横坐标均为时间采样值(samples),上半部分的纵坐标为离散系统的阶跃响应值,下半部分为离散系统的脉冲响应值。从曲线的形状来看,趋势与连续系统响应基本类似,都是趋于设定值。但由于是离散系统,每个采样时间内都有一个一阶保持器,因此曲线的外型呈台阶状。

3.2 频率响应分析

我们已经知道,直接用微分方程研究控制系统虽然可以准确的解出系统的运动函数,然而从工程角度看,这种方法既嫌不足,又嫌多余。人们所期望的工程研究方法是:计算量不应当太大,而且计算量不应微分方程的阶次升高而增加太多;容易区分出系统运动的主要因素以及各因素对系统总体动态性能的影响。以上这些要求,是直接用微分方程研究系统的方法所难以做到的。本节将要讲述的频率响应法和下一节将要讲述的根轨迹法正是满足这些要求的很好的工程研究方法。

频率响应法的基本思想是把控制系统中的各个变量看成一些信号,而这些信号又是由许多不同频率的正弦信号合成的;各个变量的运动就是系统对各个不同频率的信号的

响应的总和。

这种观察问题和处理问题的方法起源于通信科学。20 世纪 30 年代，这种观点被引进控制科学，对控制理论的发展起了强大的推动作用。它克服了直接用微分方程研究控制系统的种种困难，解决了许多理论问题和工程问题，迅速形成了分析和综合控制系统的一整套方法，即频率响应法。英国的剑桥学派又将频率响应法推广到多变量系统，因此这种方法直到今日仍然是控制理论中极为重要的基本内容。

频率响应法之所以能发挥这样的作用，是因为它具有一系列重要的优点。首先，这种方法物理意义鲜明。按照频率响应的观点，一个控制系统的运动无非是信号在一个一个环节之间依次传递的过程；每个信号又是一些不同频率的正弦信号合成的；这些不同频率的正弦信号的振幅和相角在传递过程中，依一定的函数关系变化，就产生形式多样的运动。这种观点比简单的把控制系统观念看成一个微分方程显然更容易理解，并且更能启发人们区分影响系统的主要因素和次要因素，进而考虑改善系统性能。其次，从信号传递的角度出发，可以用实验方法求出对象的数学模型，这一点在工程上价值很大，特别是对于机理复杂或机理不明而难以列写微分方程的对象，频率响应观点揭示了重要的处理方法。第三，对于手工计算来说，频率响应法的计算量小。用它分析系统的运动与直接求解系统的微分方程式相比，所需的手工计算量相差非常悬殊。第四，由于频率响应法很大一部分都采用作图，因此这种方法有很强的直观性。

当然，频率响应法不能用于对非线性系统进行全面分析，尽管它在这方面也获得了一定的成绩。因为非线性系统不满足叠加原理，所以从根本上说频率响应法不可能成为研究和设计非线性系统的得力工具。这是它主要的局限性。

对于 MATLAB 环境下控制系统的仿真和设计来说，虽然频率域分析和时间域分析在编程的工作量方面没有太大的区别，但采用频率响应的观点无疑能够更加清晰的了解系统的本质。尤其是对于系统的传递函数描述，采用频率响应法分析具有得天独厚的优势，只需将传递函数的自变量 s 换成 jω 即可。

3.2.1　频率响应

从形式上来看，频率响应法用到的频率特性函数与系统的传递函数相同，只是把自变量 s 换成 jω。对于输入量为 y，输出量为 u 的系统来说，其频率特性函数为：

$$G(j\omega)=\frac{\dot{Y}}{\dot{U}}=G(s)\Big|_{s=j\omega}$$

不过从概念上看，系统的传递函数和频率特性函数还是有一定区别的。严格地说，系统传递函数的自变量 $s=\sigma+j\omega$，其中 σ 和 ω 都是实数。事实上，频率特性函数就相当于传递函数的自变量 s 只沿复数屏幕的虚轴变化。正因为如此，传递函数和频率特性函数习惯上总是用同一字母 G 表示。

频率特性函数 G(jω)是依赖于 ω 的函数。在给定的 ω 下，它就是一个复数。习惯上用 $|G(j\omega)|$ 表示函数 G(jω)的模，它也是 ω 的函数，称为幅频特性函数。argG(jω)表示函数 G(jω)的相角，它也是 ω 的函数，称为相频特性函数。幅频特性函数表示的是正弦输出信号与正弦输入信号的振幅之比，而相频特性函数表示的是正弦输出信号相对于正弦输入信号相位的领先。因此，有：

$$\begin{cases} |G(j\omega)| = \dfrac{|\dot{Y}|}{|\dot{U}|} \\ \arg G(j\omega) = \arg\dot{Y} - \arg\dot{U} \end{cases}$$

推导系统频率特性函数的数学基础是非周期函数的傅里叶变换(Fourier Transform),相关知识在低年级的工科数学课程中已经有详细介绍,这里就不再赘述了。

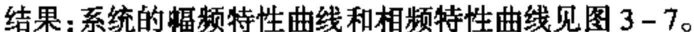
正因为系统的传递函数和频率特性函数在表达式上完全相同,因此在 MATLAB 环境下对此二者是不加以区分的。上一章处理和求解系统传递函数的语句和函数对频率特性函数同样适用。但我们在使用过程中应该注意到,传递函数的观点和系统频率响应的观点是两种不同认识控制系统的思路。在 MATLAB 环境下求解系统的幅频特性函数和相频特性函数时,经常要用到 polyval()函数。其功能是求解某一给定系数和自变量的多项式的值。其基本调用格式为:

Y = POLYVAL(P,X)

其中 P 是多项式系数,X 是自变量,Y 是求得的返回值。请看下例:

[例 3.3] 某控制系统的频率特性函数如下。试求解其幅频特性曲线和相频特性曲线,并与系统的阶跃响应曲线加以对比。

$$G(j\omega) = \frac{7(j\omega)}{(j\omega)^3 + 4(j\omega)^2 + 8(j\omega) + 9}$$



结果:系统的幅频特性曲线和相频特性曲线见图 3-7。

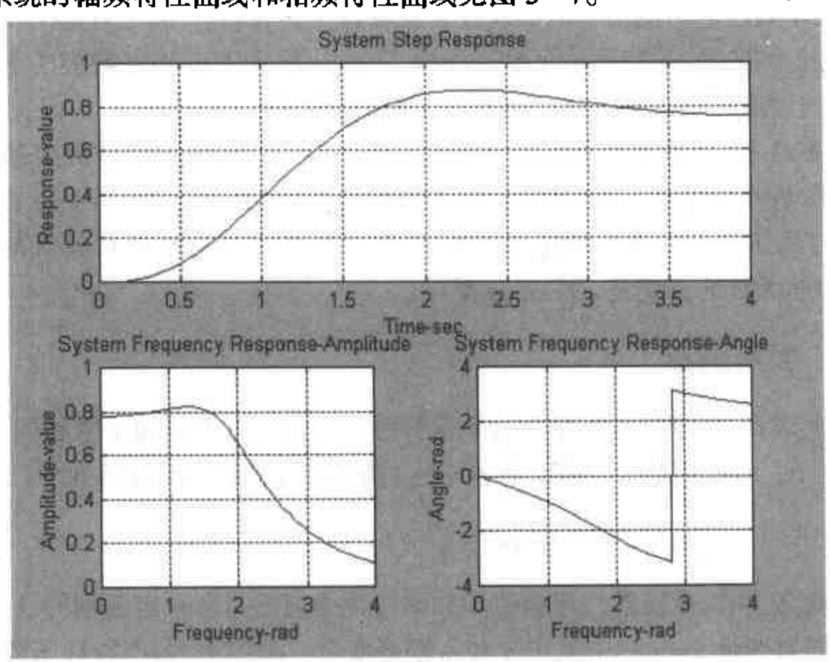


图 3-7 系统阶跃响应曲线与频率特性曲线

分析:从系统的频率特性响应曲线来看,幅频特性曲线的谐振峰比较小,低通特性很好;系统相频特性曲线的起始阶段也基本是直线,后边的跳变是因为反正切函数的特性。从系统的频率特性响应曲线上,我们还可以得出对应频率下系统的增益和相位值。对应系统的阶跃响应曲线,系统的输出量变化比较平缓,超调量比较小,振荡次数也不多,很快就达到了稳态值。因此,从系统的频率响应特性曲线我们基本上能够预测系统时间域阶

跃响应的情况,并且还可以推断任意激励函数下的响应。

求解过程:

在 MATLAB Command Window 下键入"edit"或选择"File"菜单新建 M - file,进入 MATLAB Editor/Debugger,编辑 M 文件"freqresponse.m":

```
%关闭所有窗口并清除所有内存变量
clear all;
close all;
%系统的频率特性函数描述
num=[0 0 0 7];
den=[1 4 8 9];
%仿真时间和频率初始化
```

图 3-7 的上半部分是系统的阶跃响应曲线,横坐标是时间,纵坐标是阶跃响应的值;下半部分是系统的频率响应曲线,横坐标都是频率;其中左边是相频特性曲线,右边是幅频特性曲线。

```
t=0:0.05:4;
wt=0:0.02:4;
%系统的阶跃响应
y=step(num,den,t);
%系统的频率特性
G=polyval(num,sqrt(-1)*wt)./polyval(den,sqrt(-1)*wt);
%幅频特性
mag=abs(G);
%相频特性
theta=angle(G);
%绘图
subplot(211),
plot(t,y);
title('System Step Response');
xlabel('Time-sec');
ylabel('Response-value');
grid;
subplot(223),
plot(wt,mag);
title('System Frequency Response-Amplitude');
xlabel('Frequency-rad');
ylabel('Amplitude-value');
grid;
subplot(224),
plot(wt,theta);
title('System Frequency Response-Angle');
xlabel('Frequency-rad');
ylabel('Angle-rad');
```

grid;

选择"File"菜单的"Save"选项，保存文件名为"freqresponse.m"。选择"Tools"菜单的"Run"选项或在 MATLAB Command Window 下直接键入文件名 freqresponse，在 MATLAB Command Window 下查看运行的结果。

小结：polyval()函数的自变量可以是复数 sqrt(-1)，这就给求解系统的频率特性函数带来了很大的方便。另外，本例绘制的幅频特性曲线和相频特性曲线是直接采用的直角坐标，没有采用习惯的对数或半对数坐标的波特(Bode)图。关于系统频率特性波特图的绘制，我们将在下一节介绍。

当然，上一章介绍过的 freqs()函数同样可以求解频率特性函数的响应曲线，其调用格式与参数是传递函数时一致。请看下例：

[例 3.4] 某三阶系统的传递函数如下，试根据其主导极点给出其低阶近似的频率特性函数，并比较原系统与低阶系统的阶跃响应和频率响应。

$$G(s)=\frac{750}{s^3+36s^2+205s+750}$$

结果：系统的极点为：

```
ans =  -30.0000
       -3.0000 + 4.0000i
       -3.0000 - 4.0000i
```

相应频率特性函数为：

$$G(j\omega)=\frac{25}{\left(\frac{j\omega}{30}+1\right)(j\omega+3+4j)(j\omega+-4j)}$$

可以看出，在低频段系统频率特性函数分母的第一项将非常接近于1。考虑到系统的低通特性，可以认为极点 -30 远离原点，对系统低频段影响很小，系统的主导极点为 $-3\pm4j$，对应简化的低阶模型为：

$$G(j\omega)=\frac{25}{(j\omega)^2+6(j\omega)+25}$$

原系统和低阶近似系统的阶跃响应如图 3-8 所示。

图 3-8 上半部分是原系统的阶跃响应曲线，下半部分是二阶近似系统的阶跃响应曲线。横坐标均为时间，纵坐标均为响应值。

原系统与低阶近似系统的频率响应如图 3-9 所示。其上半部分是原系统的频率响应曲线，下半部分是二阶近似系统的频率响应曲线。横坐标均为频率，纵坐标均为响应值。

分析：在实际控制系统分析过程中，寻找高阶系统的低阶近似是一种非常普遍的思路。通常采用的方法除了本例的求解系统主导极点外，还有近似零极相消等方法。所谓主导极点，是指稳定的高阶系统中离虚轴最近的一对共轭极点，附近没有零点，并且其他极点距这对极点很远或附近有零点与其相消。原高阶系统的性质与使用这对主导极点近似的二阶系统的性质大体相同。

从本例的结果图中我们也可以看出，原系统的阶跃响应和频率响应与其二阶近似系统几乎一模一样，只是在响应的起始阶段原系统的曲线更加弯曲。显然，这是因为多了一

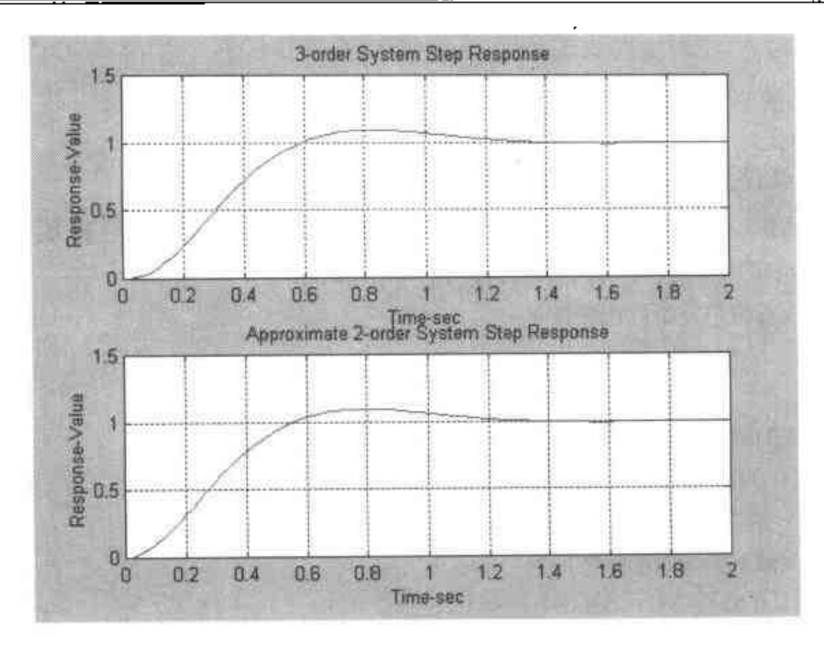


图 3-8 原系统及其低阶近似系统阶跃响应曲线比较

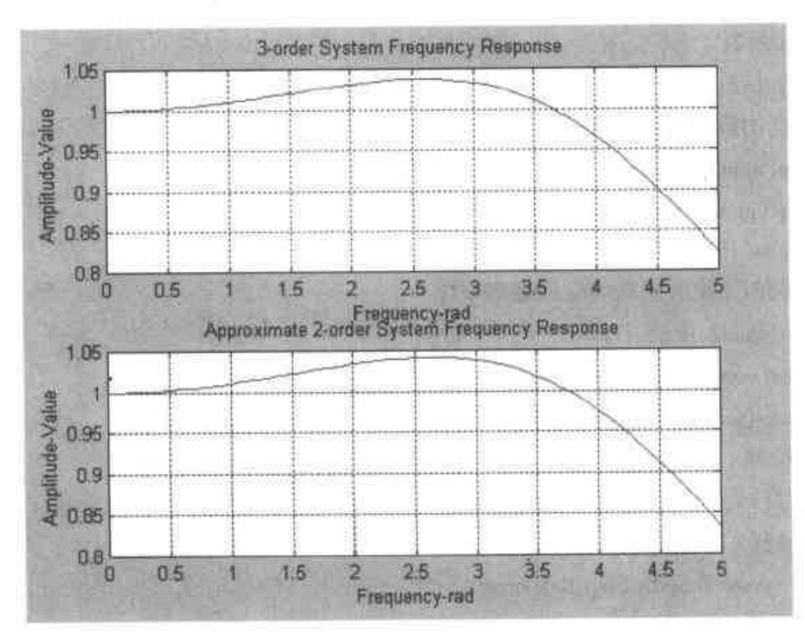


图 3-9 原系统及其低阶近似系统的频率响应曲线比较

个惯性环节的原因。在控制系统机理复杂难于分析或阶次较高的情况下,也可以先用实验的方法求出系统的一些基本参数,例如谐振峰大小和位置、截止频率、超调量和过渡过程时间等等,然后用合适的低阶系统(一般是二阶系统,因为人们对二阶系统研究得最透彻)来近似原系统。

求解过程:

本例的解题步骤分为两部分:

1. 求解系统的极点

因为本例只需求解系统的极点,所以不必调用求解零点、极点和增益的 tf2zp()函数,只需调用根据多项式系数求根的 roots()函数即可。在 MATLAB Command Window 下键

入下列语句：

```
den=[1 36 205 750];
roots(den)
```

2. 求解原系统及其低阶近似系统的阶跃响应和频率响应

在 MATLAB Command Window 下键入"edit"或选择"File"菜单新建 M - file，进入 MATLAB Editor/Debugger，编辑 M 文件"sysapprox.m"：

```
%关闭所有窗口并清除内存变量
clear all;
close all;
%原系统的频率特性函数
num1=[0 0 0 750];
den1=[1 36 205 750];
%低阶近似系统的频率特性函数
num2=[0 0 25];
den2=[1 6 25];
%仿真时间及频率初始化
t=0:0.02:2;
wt=0:0.1:5;
%原系统的阶跃响应和频率响应
y1 = step(num1,den1,t);
g1 = freqs(num1,den1,wt);
amp1=abs(g1);
%低阶近似系统的阶跃响应和频率响应
y2=step(num2,den2,t);
g2=freqs(num2,den2,wt);
amp2=abs(g2);
%图形绘制
subplot(211),
plot(t,y1);
title('3-order System Step Response');
xlabel('Time-sec');
ylabel('Response-Value');
grid;
subplot(212),
plot(t,y2);
title(' Approximate 2-order System Step Response');
xlabel('Time-sec');
ylabel('Response-Value');
grid;
subplot(211),
plot(wt,amp1);
title('3-order System Frequency Response');
```

```
xlabel('Frequency - rad');
ylabel('Amplitude - Value');
grid;
subplot(212);
plot(wt,amp2);
title('Approximate 2 - order System Frequency Response');
xlabel('Frequency - rad');
ylabel('Amplitude - Value');
grid;
```

选择"File"菜单的"Save"选项,保存文件名为"sysapprox.m"。选择"Tools"菜单的"Run"选项或在 MATLAB Command Window 下直接键入文件名 sysapprox,在 MATLAB Command Window 下查看运行的结果。

小结:本例使用了 freqs()函数代替了例 3.2 的 polyval()函数,从效果上没有什么区别。当然,调用 freqs()函数显得更简洁一些。不过使用 polyval()函数是按照原始的频率特性函数表达式的定义来计算,更容易理解频率特性函数的本质。如果调用 freqs()函数不规定返回值,就会将系统频率特性函数的数值在 MATLAB Command Window 下显示出来。不过,如果希望计算某一特定频率下系统的频率响应数值,还是调用 polyval()函数更加方便一些。

得到系统的频率响应数据之后,除了可以按照直角坐标直接绘制系统的频率特性响应曲线之外,还可以根据不同需要选取半对数坐标或极坐标绘制系统的 Bode(波特)图和 Nyquist(奈奎斯特)图。

3.2.2　Bode 图绘制

如果把频率特性函数 $G(j\omega)$的角频率 ω 和幅频特性$|G(j\omega)|$都取对数,有:

$$\begin{cases} \mu = \lg\omega \\ L(G(j\omega)) = \lg|G(j\omega)| \\ \theta(G(j\omega)) = \arg G(j\omega) \end{cases}$$

这样得到的函数 $L(G(j\omega))$和 $\theta(G(j\omega))$分别称为对数幅频特性函数和对数相频特性函数,总称对数频率特性函数,注意 θ 不取对数。以 μ 为横坐标,$L(G(j\omega))$和 $\theta(G(j\omega))$为纵坐标绘制的曲线分别被称为对数幅频特性图和对数相频特性图,统称为系统的 Bode 图。

习惯上,不管$|G(j\omega)|$的量纲和单位是什么,统一给 L 规定一个名称,称为增益;又统一给 L 规定一种单位,称为贝尔(B)。同时规定 1 贝尔等于 20 分贝尔,简称分贝。此外给 μ 也规定一种单位,称为十倍频程。不过十倍频程只用来度量 μ 的两个值之差,而不用来度量 μ 值本身。

用 Bode 图表示对象的性质优点很多。首先,可以展宽视野。在对数坐标图上,系统的低频、中频和高频段的性质都可以以适当的比例清晰的展现出来。而在直角坐标图上,只有中频段表现的比较充分,低频段和高频段都不能得到充分反映。其次,曲线形状比较简单,易于手工绘制。这在缺乏计算机辅助设计软件的场合是非常重要的。第三,由于对数运算将四则运算的乘除化为加减,因此,可以用简单叠加的方式得到方框图串联形式连

接的系统的总体的 Bode 图。

除了根据系统频率响应数据直接绘制 Bode 图外，MATLAB 还提供了一条函数 bode()可以直接求解和绘制系统的 Bode 图。其基本调用格式为：

(MAG,PHASE) = BODE(SYS,...,W)

其中 SYS 是系统的频率特性函数或状态空间描述(MIMO 系统要指明是对第几个输入量)；W 是指定的角频率向量，也可以不加指定而由 MATLAB 自己给出；MAG 和 PHASE 是返回的系统开环幅频特性和相频特性，如果没有返回值的话 MATLAB 就在屏幕上绘制出缺省参数的系统 Bode 图。

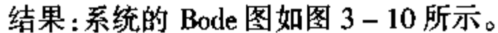
MATLAB 还提供了一条函数 margin()，可以求解系统的增益裕量和相角裕量，其基本调用格式为：

[Gm,Pm,Wcg,Wcp] = MARGIN(SYS)

其中 Gm、Pm、Wcg 和 Wcm 分别是系统的增益裕量、相角裕量及其对应的角频率。

[例 3.5] 某控制系统的开环传递函数如下。试绘制其 Bode 图和增益裕量以及相角裕量，并与系统的闭环阶跃响应曲线比较。

$$G(s)=\frac{0.86}{s(0.36s+1)(0.3906s^2+0.75s+1)}$$



结果：系统的 Bode 图如图 3－10 所示。

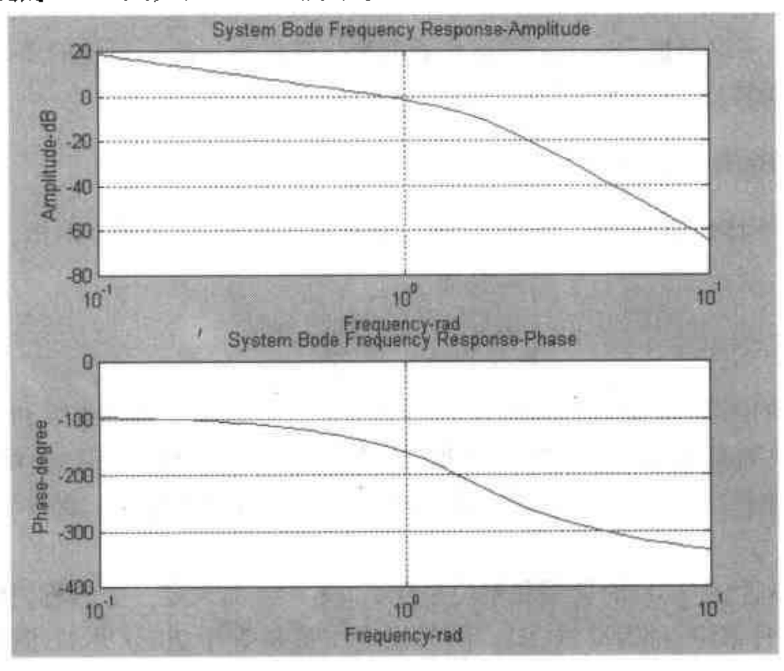


图 3－10 四阶系统 Bode 图

图 3－10 的横坐标均为频率弧度值，注意其刻度是以 10 的幂的形式标注的。其中上半部分是系统的相频响应曲线，纵坐标为增益的 dB 值，下半部分为系统的幅频响应曲线，纵坐标为相角的角度值。

闭环系统阶跃响应曲线如图 3－11 所示。其横坐标为时间，纵坐标为系统的阶跃响应值。

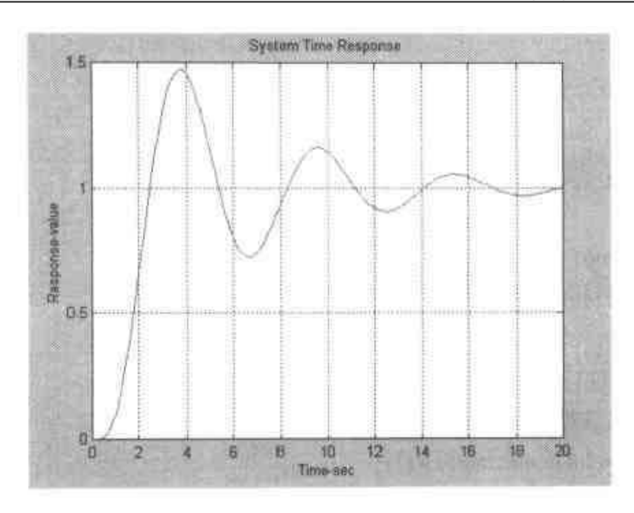


图 3-11 四阶系统闭环阶跃响应曲线

系统的增益裕量和相角裕量为

```
System Gain Margin and its associated frequency are:
Gm = 1.5791
Wcg = 1.2304
System Phase Margin and its associated frequency are:
Pm = 30.8985
Wcm = 0.8561
```

分析:从系统的 Bode 图上可以看出,低频段对数幅频特性曲线的斜率约为 -20dB 每十倍频程,这是因为系统的开环传递函数有积分项 1/s。相应的对数相频特性曲线也是从 -90°开始的,并且因为是四阶系统,最后趋于 -360°。

系统的增益裕量约为 1.5791/dB,相角裕量约为 30.8985°,从系统 Bode 图的 0dB 线和 -180°线也可以大致得到这个结果。从稳定裕量上看,系统是稳定的,系统的闭环阶跃响应曲线也可以验证这一点。不过系统的截止角频率较小,约 0.8561。因此从系统的闭环阶跃响应曲线上来看,系统的过渡过程时间较长(约 20s),超调量较大(约 50%),振荡次数也较多。

对于这样的高阶系统,仅仅用二阶系统近似一般是不能满足要求的,一般都要作串联校正或反馈校正,将系统的主导极点配置到合适的位置上来。这方面相关的 MATLAB 实现,我们将在下一章介绍。

求解过程:

在 MATLAB Command Window 下键入"edit"或选择"File"菜单新建 M - file,进入 MATLAB Editor/Debugger,编辑 M 文件"sysbode.m"(注意不要直接取名为 bode.m,否则会冲掉 MATLAB 系统原有的 bode.m 文件):

```
%关闭所有图形窗口并清除内存变量
close all;
clear all;
%系统开环传递函数描述
numo=[0 0 0 0 0.86];
den1=[1 0];
```

```
den2 = [0.36,1];
den3 = [0.3906 0.75 1];
deno = conv(den1,conv(den2,den3));
%系统闭环传递函数描述
numc = numo;
denc = deno + numo;
% 系统仿真时间及频率初始化
t = 0:0.1:20;
wt = logspace(-1,1);
%系统频率响应数据
[mag,phase] = bode(numo,deno,wt);
%系统增益裕量及相角裕量
[Gm,Pm,Wcg,Wcm] = margin(numo,deno);
%显示结果
disp('System Gain Margin and its associated frequency are:');
Gm
Wcg
disp('System Phase Margin and its associated frequency are:');
Pm
Wcm
%系统的闭环阶跃响应
y = step(numc,denc,t);
%图形绘制
subplot(211),
%对数幅频特性函数
amp = 20 * log10(mag);
semilogx(wt,amp)
title('System Bode Frequency Response - Amplitude');
xlabel('Frequency - rad');
ylabel('Amplitude - dB');
grid;
subplot(212),
semilogx(wt,phase);
title('System Bode Frequency Response - Phase');
xlabel('Frequency - rad');
ylabel('Phase - degree');
grid;
plot(t,y);
title('System Time Response');
xlabel('Time - sec');
ylabel('Response - value');
grid;
```

选择“File”菜单的“Save”选项，保存文件名为“sysbode.m”。选择“Tools”菜单的

“Run”选项或在 MATLAB Command Window 下直接键入文件名 sysbode，在 MATLAB Command Window 下查看运行的结果。

小结：本例用到了一个新的 MATLAB 函数 conv()，其功能是求解两个向量的卷积或求解两个多项式的乘积。熟悉信号处理的读者都清楚，这两者在数学上是一致的。其基本调用格式为 C = CONV(A, B)，其中 A 和 B 是两个向量，可以代表待卷积的序列，也可以代表多项式的系数(不必限制一定是降幂排列，但两者的排列方式必须一致)；C 是返回的向量，维数是 LENGTH(A) + LENGTH(B) - 1。对于调用该函数求解多项式乘积来说，需要注意的是，此函数只能计算两个多项式的乘积。如果希望计算 n 个多项式的乘积，或者调用 conv()函数 n - 1 次，或者自己编写相应的 M 函数。

本例在绘制 Bode 图和求解稳定裕量时用的是开环频率特性函数，而绘制阶跃响应曲线时用的是闭环频率特性函数。这是因为最初系统的闭环频率特性图是难于绘制的，一般是通过开环频率特性去估计。MATLAB 继承了这一做法，因此 bode()函数和 margin()函数都是为开环频率特性函数编写的。当然读者也可以编写求解闭环频率特性函数 Bode 图和稳定裕量的函数。

控制理论中还有一种对数频率响应图称为对数幅相特性图。其含义是在一定的频率范围内，以相角 θ 作为横坐标，以对数幅值 L 作为纵坐标，以角频率 ω 作为参变量绘制的频率特性图，也称为 Nichols 图。MATLAB 同样提供了绘制此对数幅相特性图的函数 nichols()，其调用格式与 bode()函数完全一数。例如执行完例 3.3 的 sysbode.m 文件之后，在 MATLAB Command Window 下键入：

```
nichols(numo,deno)
```

可得该系统的对数幅相特性曲线如图 3 - 12 所示。其横坐标为开环系统的相角，单位是度；纵坐标为系统的开环增益，单位是 dB。

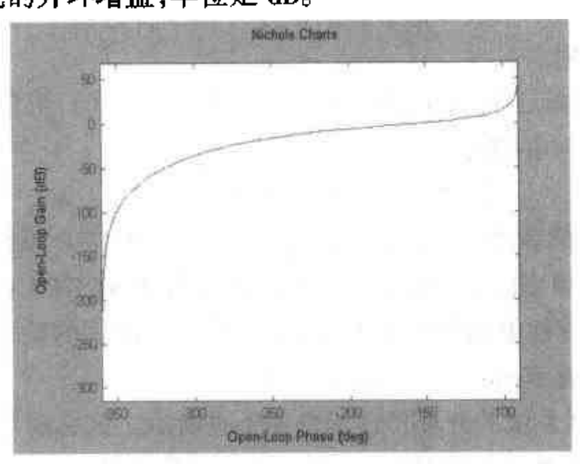


图 3 - 12　系统的 Nichols 曲线

从系统的对数幅相特性曲线中同样可以读出系统的增益裕量和相角裕量，不过相应的角频率 ω 就需要重新计算了。

3.2.3　Nyquist 图绘制

按照频率特性函数的本意，频率 ω 是正实数，但是如果不管物理意义，只把它看作是 ω 的一个函数，那么我们也可以考察当 ω 为负数时的频率特性函数的图像。一切集总参

数对象的传递函数 G(s)都是 s 的有理函数,所以它们的频率特性函数都是 $j\omega$ 的有理函数。因此容易理解,对于任一给定的频率 ω,$G(j\omega)$ 和 $G(-j\omega)$ 必定是互为共轭的一对复数。它们在复平面上的位置是关于实轴对称的。

习惯上,我们称频率特性函数 $G(j\omega)$ 在复平面的图像为系统的极坐标频率特性图或 Nyquist 图。

为了方便,有时要研究 $G(j\omega)$ 的倒数,称为逆频率特性函数,定义为:

$$\hat{G}(j\omega) = \frac{1}{G(j\omega)}$$

显然,逆幅频和逆相频特性函数与幅频和相频特性函数有如下的关系:

$$\begin{cases} |\hat{G}(j\omega)| = \dfrac{1}{|G(j\omega)|} \\ \arg\hat{G}(j\omega) = -\arg G(j\omega) \end{cases}$$

它们的逆幅频和逆相频特性函数图像就称为逆 Nyquist 图,在多变量频域设计的对角化方面有一定的应用。

通过系统的开环频率特性函数 Nyquist 图,还可以判断系统的稳定性,这就是著名的 Nyquist 稳定判据。Nyquist 稳定判据叙述如下:

如果系统的开环传递函数 G(s)在右半 s 平面有 p 个极点,则闭环系统稳定的充分必要条件是:当 ω 从 $-\infty$ 连续的变化到 $+\infty$ 时,开环频率特性函数 $G(j\omega)$ 的 Nyquist 图逆时针方向包围复平面上的 -1 点 p 圈。

应用 Nyquist 稳定判据研究控制系统的稳定性有很多优点:首先,它主要靠作图,手工计算量很小。其次,它不仅能回答闭环系统是否稳定,而且还可以得到系统接近于不稳定的程度。更进一步,这个判据往往还可以提示改善系统稳定性的办法。除此之外,在实际应用中这个判据可以不要求知道系统的微分方程或传递函数,而只要依靠实验测出其开环频率特性即可应用。

MATLAB 控制系统工具箱中提供了一条函数 nyquist(),可以用来求解或绘制系统的 Nyquist 图。其基本调用格式为:

```
[RE,IM] = NYQUIST(SYS,...,W)
```

其中 SYS 是系统的频率特性函数或状态空间描述(MIMO 系统要指明是对第几个输入量);W 是指定的角频率向量,也可以不加指定而由 MATLAB 自己给出;RE 和 IM 是返回的系统开环频率特性依的实部和虚部,如果没有返回像的话,MATLAB 就在屏幕上绘制出缺省参数的系统 Nyquist 图。请看下例:

[例 3.6] 某控制系统的开环传递函数如下页所示。试绘制当 K = 2 和 K = 20 时系统的 Nyquist 图。根据 Nyquist 稳定判据判断系统的稳定性,并绘制系统的闭环阶跃响应曲线验证关于稳定性的结论。

$$G(s) = \frac{K}{s(s+1)(0.1s+1)}$$

结果:系统在 K = 2 时的 Nyquist 图如图 3-13 所示。其横坐标为系统频率响应的实部,纵坐标为虚部。

K = 20 时系统的 Nyquist 曲线如图 3-14 所示。同样,其横坐标为系统频率响应的实部,纵坐标为虚部。

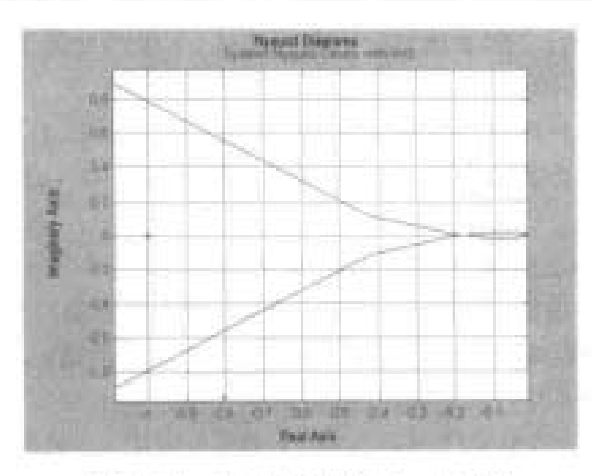


图 3-13　K = 2 时系统的 Nyquist 曲线

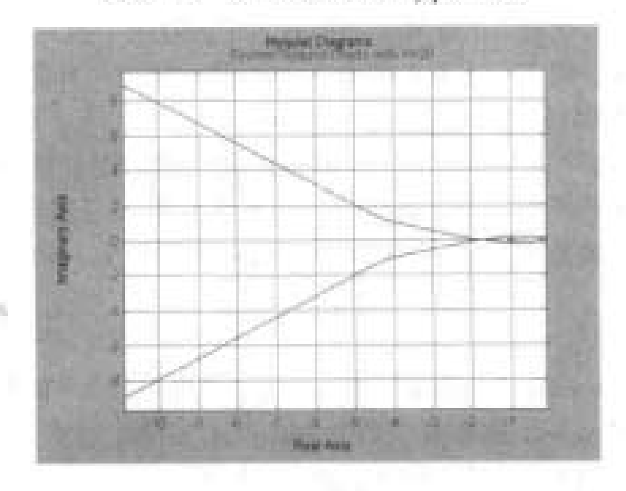


图 3-14　K = 20 时系统的 Nyquist 曲线



K = 2 和 K = 20 系统的闭环阶跃响应曲线见图 3-15。

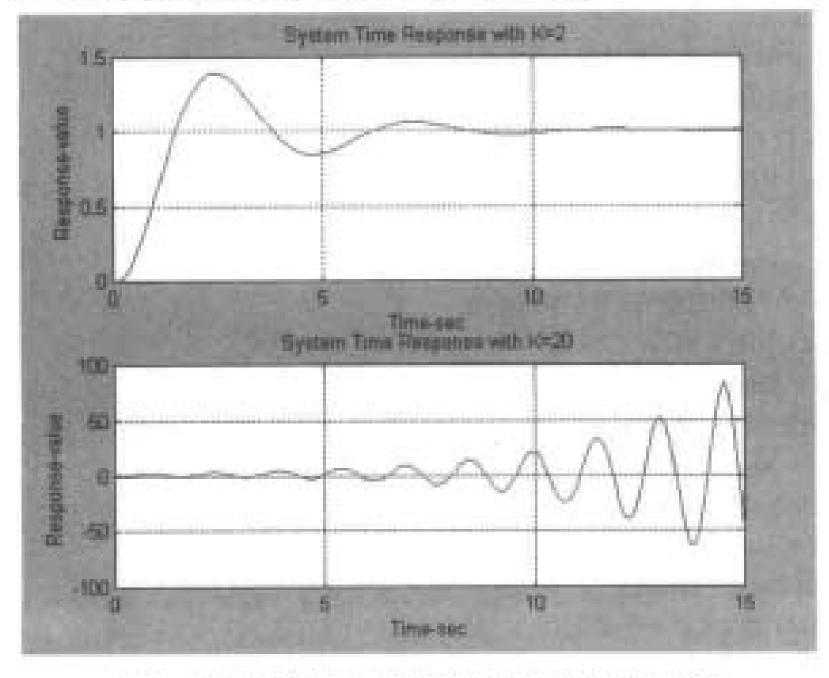


图 3-15　K = 2 和 K = 20 时系统的闭环阶跃响应曲线

其中上半部分是 K = 2 时系统的阶跃响应曲线，下半部分是 K = 20 时系统的阶跃响应曲线。其横坐标均为时间，纵坐标均为系统的阶跃响应值。

分析：根据系统的开环传递函数表达式，系统在复平面右半平面的极点数为 0 因此，该系统稳定的充分必要条件为：系统开环频率特性函数 G(jω) 的 Nyquist 曲线不包含复平面的(-1,0)点。

从图 3-13 的 K = 2 时系统 Nyquist 图来看，图中用"+"号表示的复平面的(-1,0)点远离系统的 Nyquist 曲线，因此，系统的闭环时间域响应是稳定的。图 3-15 上半部分 K = 2 时系统的闭环阶跃响应曲线也验证了这种说法，系统的输出量在振荡两次后大约从 10s 开始进入稳态值 5%左右的区间内，并最终趋于设定值 1。

从图 3-14 的 K = 20 时系统 Nyquist 图来看，图中用"+"号表示的复平面的(-1,0)点被系统的 Nyquist 曲线包围。当 ω 从 -∞ 连续的变化到 +∞ 时，"+"号被系统的 Nyquist 曲线包围两圈，说明系统在复平面的右半平面有两个极点。因此，系统的闭环时间域响应是不稳定的。图 3-15 下半部分 K = 20 时系统的闭环阶跃响应曲线同样验证了这种说法，系统的输出量增幅振荡，最终趋于∞。

注意：这里用的是"验证"而非"证明"，因为直观的定性观察并不能作为理论推导的结果。因为本例系统的开环频率特性函数是 jω 的有理函数，所以实轴以上的部分应该与实轴以下的部分关于实轴对称，图 3-13 和图 3-14 的系统 Nyquist 曲线都验证了这种说法。

求解过程：

在 MATLAB Command Window 下键入"edit"或选择"File"菜单新建 M-file，进入 MATLAB Editor/Debugger，编辑 M 文件"sysnyquist.m"（注意不要直接取名为 nyquist.m，否则会冲掉 MATLAB 系统原有的 nyquist.m 文件）：

```
%关闭所有图象窗口并清除内存变量
close all;
clear all;
% K = 2 时系统的开环频率特性函数
numo1 = [0 0 0 2];
den1 = [1 0];
den2 = [1 1];
den3 = [0.1 1];
deno1 = conv(den1,conv(den2,den3));
% K = 20 时系统的开环频率特性函数
numo2 = [0 0 0 20];
deno2 = deno1;
%系统的闭环频率特性函数
numc1 = numo1;
denc1 = deno1 + numo1;
numc2 = numo2;
denc2 = deno2 + numo2;
%仿真时间及角频率初始化
t = 0:0.1:15;
```

```
wt = -10:1:10;
%K = 2 时系统的开环 Nyquist 曲线
nyquist(numo1,deno1,wt);
title('System Nyquist Charts with K = 2')
grid;
%K = 20 时系统的开环 Nyquist 曲线
nyquist(numo2,deno2,wt);
title('System Nyquist Charts with K = 20')
grid;
%系统的闭环阶跃响应
y1 = step(numc1,denc1,t);
y2 = step(numc2,denc2,t);
%绘图
subplot(211),
plot(t,y1);
title('System Time Response with K = 2');
xlabel('Time - sec');
ylabel('Response - value');
grid;
subplot(212),
plot(t,y2);
title('System Time Response with K = 20');
xlabel('Time - sec');
ylabel('Response - value');
grid;
```

选择"File"菜单的"Save"选项,保存文件名为"sysnyquist.m"。选择"Tools"菜单的"Run"选项或在 MATLAB Command Window 下直接键入文件名 sysnyquist,在 MATLAB Command Window 下查看运行的结果。

小结:使用 nyquist()函数绘制系统的开环 Nyquist 曲线时,尤其要注意函数的调用格式和参数设置。因为应用 Nyquist 稳定判据时关键就在系统 nyquist 曲线在复平面(-1,0)点附近的情况,所以在选择参变量 ω 时就既要能反映曲线在(-1,0)点附近的变化情况,又要能看出曲线在 ω 趋于 ∞ 时的变化趋势。因此,对于不同的系统 ω 的选择是不同的。当然,使用系统给定的 ω 也是一种办法,但往往不能奏效。另外,如果使用 nyquist()而非 plot()函数绘图,MATLAB 自动在图中(-1,0)点标出一个"+"号,方便使用者应用 Nyquist 稳定判据。

3.2.4　离散系统的频率响应

离散控制系统又称采样控制系统,是一种采用断续方式控制的系统。一般这种系统中都含有通常称为采样器的专门开关装置,定时开启和关闭。近几十年由于数字式计算机的飞速发展,离散控制系统在现代工业控制中获得了非常广泛的应用。数字控制器或计算机在控制速度、控制精度以及性能价格比都比模拟控制器有着明显的优越性,并且数

字控制器还有量化的通用性。事实上,不管是连续系统还是离散系统,使用数字式计算机进行仿真时都要化成离散时间系统。

离散控制系统的理论基础是采样定理,数学工具是 Z 变换。相信读者都已经在相关的课程中有所接触,这里就不再赘述了。

在本书的 3.1 节里曾经介绍过 MATLAB 环境下离散控制系统时间域仿真的函数 dstep()和 dimpulse(),频率域的情形也基本类似,同样是连续系统的频率响应函数前边加 d 就成了相应的离散系统的频率域分析函数。例如 dbode()函数、dnyquist()函数和 dnichols()函数,分别是求取离散系统的 Bode 图、Nyquist 图和 Nichols 图。这些函数的调用格式和参数设置和相应的连续系统函数也大体相同,不过需要注意的是,这些函数的参数中多了采样时间 Ts 这样一个必选项,其功能是设定该离散相同仿真的时间间隔。下面以一个简单的例子使读者熟悉一下这些函数的用法。

[例 3.7] 某控制系统传递函数如下,在其输入部分加一个 Ts = 2s 的采样器,试求其离散系统模型并绘制系统的 Bode 图、Nyquist 图和 Nichols 图。

$$G(s) = \frac{2}{10S+1}, \frac{1.2}{2s+1}$$

结果:离散系统的 Bode 图见图 3 - 16 所示。

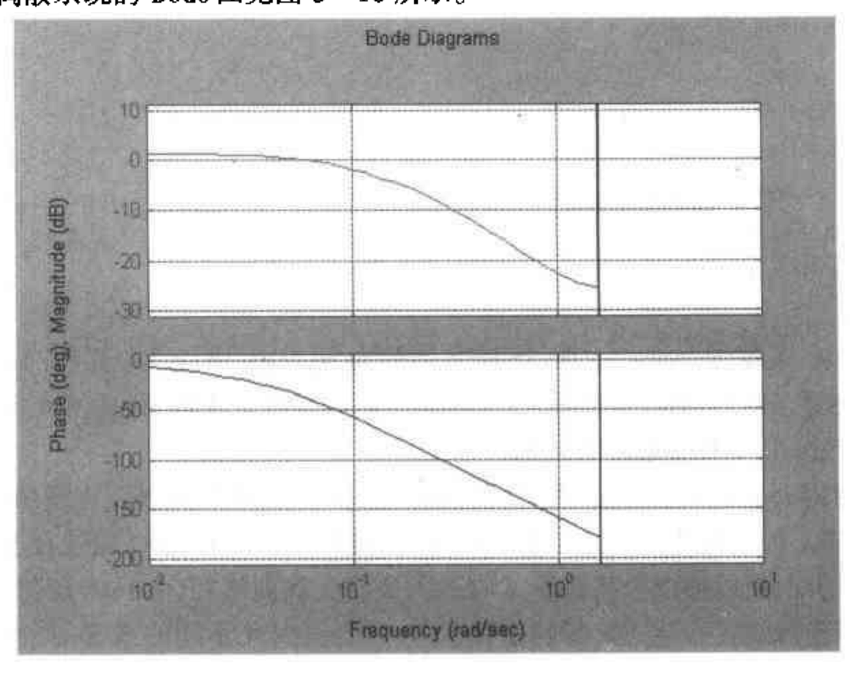


图 3 - 16 离散系统 Bode 图

其中上半部分是系统的相频特性曲线,下半部分是系统的幅频特性曲线。横坐标均为频率,单位为 rad/s;纵坐标分别是增益、单位 dB 和相角,单位(°)。

离散系统的 Nyquist 图和 Nichols 图见图 3 - 17。其中上半部分是离散系统的 Nyquist 图,横坐标是系统频率响应实部,纵坐标为频率响应的虚部;下半部分是离散系统的 Nichols 图,横坐标为开环系统的相角,单位是(°);纵坐标为系统的开环增益,单位是 dB。

离散系统模型为:

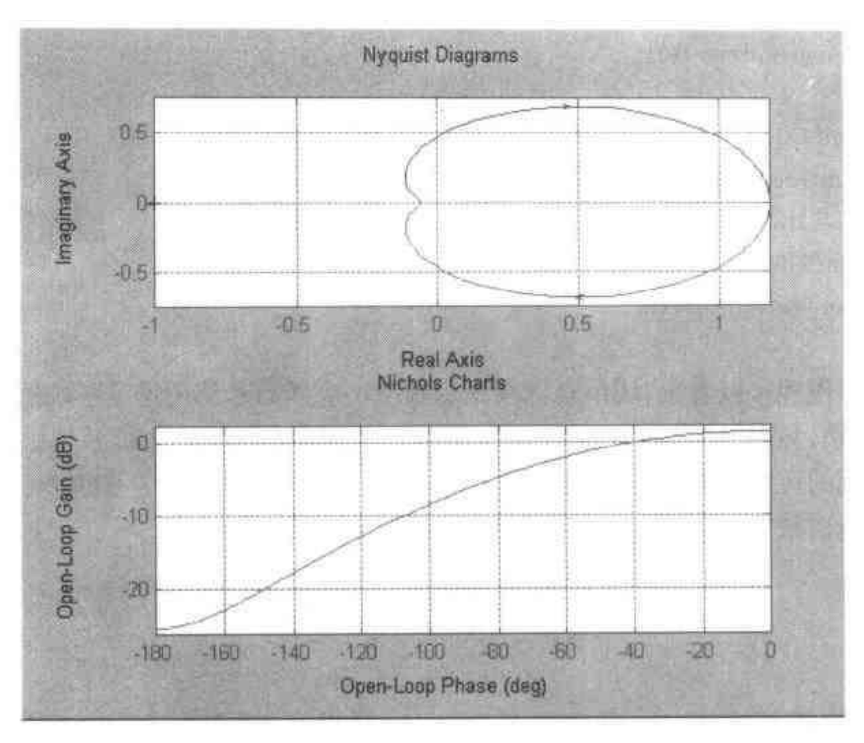


图 3－17　离散系统的 Nyquist 图和 Nichols 图

$$H(z)=\frac{0.135z}{(z-0.819)(z-0.368)}$$

求解过程：

本例的解题步骤分为两部分。

1. 求解离散系统的脉冲传递函数。

由：
$$G(s)=\frac{2}{10s+1}\cdot\frac{1.2}{2s+1}=\frac{3}{10s+1}-\frac{0.6}{2s+1}$$

取拉普拉斯逆变换，然后在取 Z 变换，并代入 Ts = 2，得：

$$H(z)=\frac{0.3z}{z-e^{-0.1T}}-\frac{0.3z}{z-e^{-0.5T}}=\frac{0.135z}{(z-0.819)(z-0.368)}$$

上述过程可以用前边介绍过的 residue()函数得到。

2. 编写求解频率响应曲线的 M 文件

因为本例只是示意函数的用法和效果，并非根据曲线分析系统，所以程序编写的比较简单。可以在 MATLAB Editor/Debugger 下编辑存盘，也可以在 MATLAB Command Window 下直接键入执行：

```
%系统的开环模型
numo = [0 0.135 0];
den1 = [1  -0.819];
den2 = [1  -0.368];
deno = conv(den1,den2);
%采样时间
Ts = 2;
%系统 Bode 图、Nyquist 图和 Nichols 图绘制
```

```
dbode(numo,deno,Ts);
grid;
subplot(211),
dnyquist(numo,deno,Ts);
grid;
subplot(212),
dnichols(numo,deno,Ts);
grid;
```

小结：本例程可以看出，调用离散系统的频率响应函数和调用连续系统的相应函数没什么太大区别，只不过注意一下采样时间 Ts 而已。当然，同样的连续系统采用不同的采样时间，得到的可能是完全不同的离散系统。并且有可能把稳定的系统变为不稳定的。因此，在进行离散系统分析时，尤其要注意采样时间的选择。

3.3 根轨迹的绘制

根轨迹法是 W.B.Evans 于 1948 年提出的一种求解变换特征方程根的简便的图解的方法，在工程上获得了广泛的应用。它根据系统开环传递函数极点和零点的分布，依照一些简单的规则，用作图的方法求出闭环极点的分布，避免了复杂的数学计算。有些读者可能会问，既然使用 MATLAB 控制系统工具箱提供的函数可以很方便的求出系统的闭环传递函数及其零点、极点和增益，那么类似根轨迹法这种间接分析系统闭环传递函数的方法不就没有生命力了吗？其实不是这样的。一方面，应用根轨迹法可以很容易地从图上看出来系统中某个参数的变化会对系统的闭环极点产生什么影响，进而如何影响系统的动态性能；另一方面，完整的根轨迹图形也是控制系统设计和校正的一种得力的辅助工具和检验手段。另外，应用根轨迹法的手工计算量很小，结果也比较可靠，在没有控制系统计算机辅助设计软件的情况下未尝不是一种很好的选择。下边简要介绍一下根轨迹的简便特性及绘制规则。

所谓根轨迹，就是指系统的开环传递函数增益变化时其变化传递函数的极点在复平面上变化的轨迹。

设系统的开环传递函数为：

$$G_0(s)=\frac{KN(s)}{D(s)}$$

其中 N(s)和 D(s)分布是 s 的 m 次和 n 次多项式，且 $n\geqslant m$。则闭环特征方程为：

$$KN(s)+D(s)=0$$

显然，当 N(s)和 D(s)没有公因子时，这个方程与下面的方程 $G_0(s)=-1$ 同根。

因此，可以将系统的开环传递函数写成下述形式：

$$G_0(s)=\frac{KN(s)}{D(s)}=\frac{K(s-z_1)\cdots(s-z_m)}{(s-p_1)\cdots(s-P_n)}$$

将上式两端分别取模和取角，就得到绘制根轨迹的模条件和角条件分别为：

$$K\frac{\prod_{i=1}^{m}|s-z_i|}{\prod_{j=1}^{n}|s-P_j|}=1$$

$$\sum_{i=1}^{n}\arg(s-z_i)-\sum_{j=1}^{n}\arg(s-P_j)=(2k+1)\pi$$

以上就是绘制闭环传递函数根轨迹的理论基础。人们根据绘制根轨迹的模条件和角条件总结了以下几条根轨迹绘制的基本规则：

(1) 根轨迹的对称性：根轨迹关于实轴对称。

(2) 根轨迹的分支数、起点和终点：对 $n>m$，根轨迹共有 n 条分支。根轨迹的起点是开环极点，有 m 条根轨迹的终点是开环零点，其余 $n-m$ 条根轨迹的终点在无穷远点，称为无穷远开环零点。

(3) 根轨迹在实轴上的分布：实轴上的某一段属于系统闭环传递函数的根轨迹，当且仅当其右侧开环传递函数的实极点和实零点数目之和为奇数。

(4) 根轨迹的渐近线：根轨迹共有 $n-m$ 条渐近线，都是从实轴上的共同交点向外辐射，辐射角(亦称渐近线的倾角)为：

$$\theta=\frac{2k+1}{n-m}\pi$$

(5) 根轨迹的分离点及会合点：如果实轴上相邻两极点间的线段属于根轨迹，则它们之间必有分离点。同理，如果实轴上相邻两零点间的线段属于根轨迹，则它们之间必有会合点。

(6) 实轴上分离点的分离角及会合点的会合角：实轴上分离点的分离角恒为 ±90°。同理，实轴上会合点的会合角恒为 ±90°。

(7) 根轨迹在复极点(或复零点)处的出射角(或入射角)：出射角就是从复极点 p 出发的根轨迹切线的方向角，其值为：出射角 = Σ[各零点指向本极点方向的方向角] − Σ[其他极点指向本极点的方向角] + 反向。

(8) 根轨迹与虚轴的交点及临界根轨迹的增益值：将 $s=j\omega$ 代入特征方程可求得 ω 和 K，即根轨迹与虚轴交点的坐标及交点所对应的临界根轨迹增益值。

通过以上这些规则，一般都可以绘制出比较满意的根轨迹示意图。当然，MATLAB 提供了一条函数 rlocus()，可以通过系统的开环传递函数绘制其闭环的根轨迹图。不必再手工绘制。其基本调用格式为：

R = RLOCUS(SYS, K)

其中 SYS 是系统的开环传递函数描述，K 是系统增益向量，也可以缺省并使用 MATLAB 提供的向量；R 是返回的根轨迹数据。如果不设返回值，MATLAB 就自动绘制系统的闭环传递函数的根轨迹。请看下例：

[例 3.8] 某控制系统的开环传递函数如下，试绘制系统的闭环根轨迹。寻找系统临界稳定时的增益 K，并绘制此时的系统阶跃响应作为验证。

$$G_0(s)=\frac{K}{(s+1)(s+3-j)(s+3+j1)}$$

结果：系统的闭环传递函数根轨迹如图 3－18 所示。

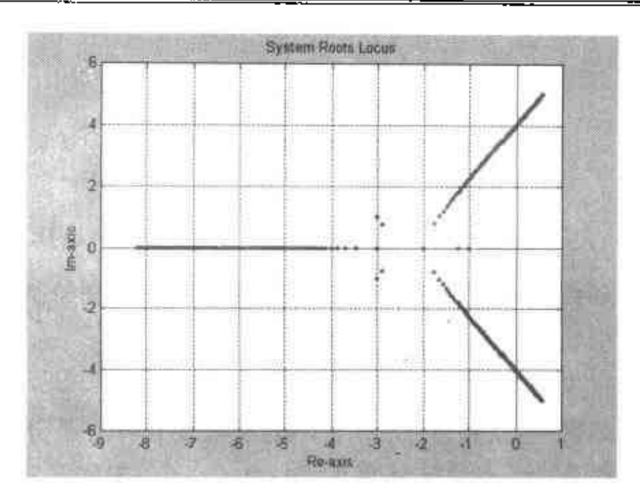


图 3 - 18 系统闭环根轨迹图

其横纵坐标轴在图中以蓝色实线描出,分别是系统频率响应的实部和虚部。图中带间断点的粗实线就是系统的根轨迹。

系统的临界稳定增益为:

System critical stable - point is:

K(i) = 102

临界稳定时系统的阶跃响应曲线如图 3 - 19 所示。其中横坐标为时间,纵坐标为系统的响应值。

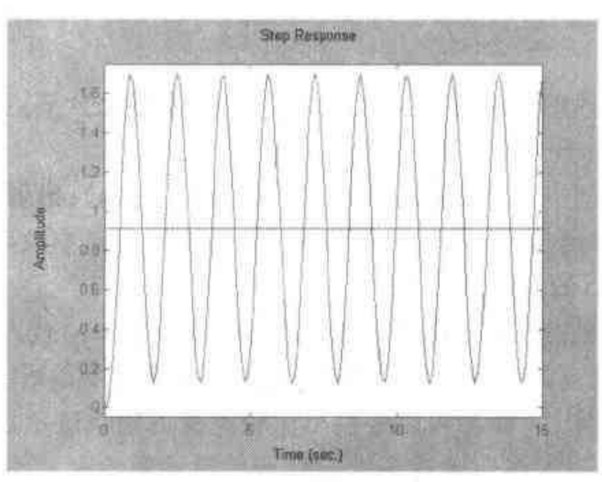


图 3 - 19 系统临界稳定时的响应曲线

分析:从图 3 - 18 中可以看出,系统的闭环根轨迹有 3 条分支,对应系统的极点与零点之差也为 3。出射角方向为 180° ÷ 3 = 60°。系统的临界稳定增益为 102,此时系统的阶跃响应为等幅振荡(见图 3 - 19)。

求解过程:

在 MATLAB Command Window 下键入"edit"或选择"File"菜单新建 M - file,进入 MATLAB Editor/Debugger,编辑 M 文件"sysrlocus.m"(注意不要直接取名为 rlocus.m,否则会冲掉 MATLAB 系统原有的 rlocus.m 文件):

```
%系统开环传递函数
```

```
num = 1;
den1 = [1 1];
den2 = [1 3 - sqrt( - 1)];
den3 = [1 3 + sqrt( - 1)];
den = conv(den1,conv(den2,den3));
%增益相邻初始化
K = 0:1:200;
%求解根轨迹数据
r = rlocus(num,den,K);
%寻找临界稳定点
[m,n] = size(r);
for i = 1:m
    if real(r(i,2)) > 0
        break;
    end
end
%显示结果
disp('System critical stable - point is:')
K(i)
%根轨迹绘图
x = - 9:0.1:1;
y = - 6:0.1:6;
tx = zeros(1,length(x));
ty = zeros(1,length(y));
plot(r,'.');
line(x,tx);
line(ty,y);
title('System Roots Locus');
xlabel('Re - axis');
ylabel('Im - axis');
grid;
%临界稳定点系统闭环传递函数
numc = zeros(1,length(den));
numc(length(den)) = K(i);
denc = den + numc;
%临界稳定时系统的阶跃响应曲线
step(numc,denc);
```

选择“File”菜单的“Save”选项，保存文件名为“sysrlocus.m”。选择“Tools”菜单的“Run”选项或在 MATLAB Command Window 下直接键入文件名 sysrlocus，在 MATLAB Command Window 下查看运行的结果。

小结：本例使用了一个新函数 line()，其功能是在指定的位置画直线。

3.4 小　　结

到此为止，本书对 MATLAB 在控制系统的时频分析中的应用就告一段落。在本章中我们主要介绍了 MATLAB 环境下控制系统时域响应分析、频率响应分析，包括 Bode 图绘制、Nyquist 图绘制、离散控制系统频率响应，还有根轨迹的绘制等等。对于以上各部分内容，其关键有两点：一是求解系统的响应曲线，对系统的性能作出评价；二是根据各种间接的方法判断系统的稳定性。

在本章控制系统时频分析的基础之上，下一章我们将讲述有关 MATLAB 环境下控制系统校正的内容。

第四章　传递函数模型控制系统校正

在前面几章里，我们讨论了控制系统的两种工程分析方法：时间域方法和频率域方法（根轨迹方法也是一种频率方法）。利用这些方法我们能够在系统结构和参数已确定的条件下，计算和估计它们的性能。这个问题通常被称作系统的分析问题。但在工程实际中常常提出相反的要求，就是说，被控对象是已知的，性能指标是预先给定的，要求设计者选择合适的结构和参数，使控制器与被控对象组成一个性能满足要求的系统。这个问题叫做系统的综合。可见，综合的目的就是在系统中引入合适的附加装置，使原有系统的缺点得到校正，从而满足一定的性能指标。引入的附加装置通常称为校正装置。所以系统的综合问题就是选择校正装置接入的位置以及它的结构参数的问题。有时也笼统的把系统的综合称为系统的校正。本章所要介绍的，主要是 MATLAB 在单变量控制系统校正中的一些应用，其重点的函数都包含在控制系统工具箱中。

对于单变量系统来说，校正装置接入系统的主要形式有两种：一种是校正装置与被校正对象相串联，如图 4－1(a)所示，这种校正方式称为串联校正。另一种形式是从被校正对象中引出反馈信号，与被校正对象或其一部分构成局部反馈回路，并在局部反馈回路内设置校正装置。这种校正方式称为局部反馈校正或“并联校正”，如图 4－1(b)所示。本章将重点讲述有关串联校正的内容。

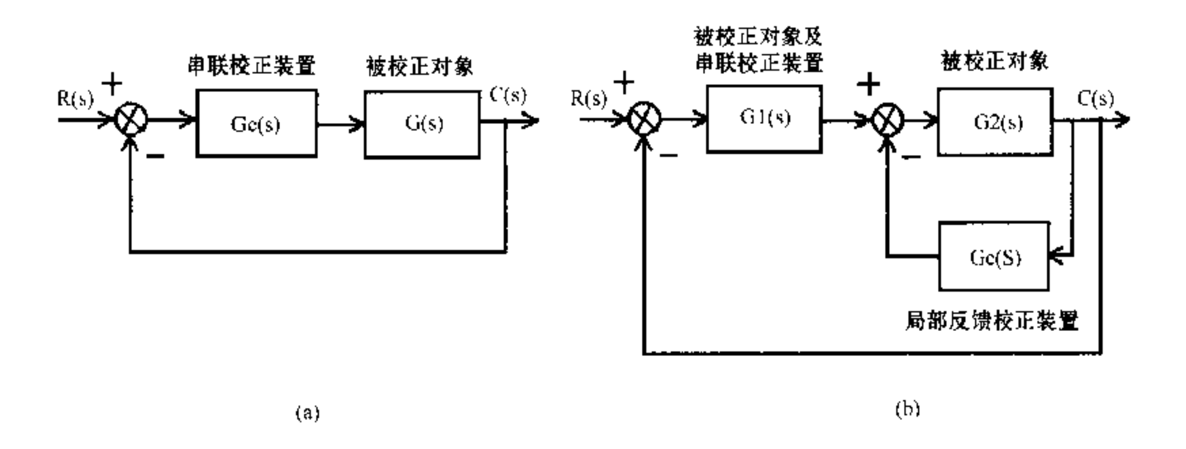


图 4－1　控制系统校正的两种主要形式

MATLAB 控制系统工具箱中提供了一些用于串联校正和局部反馈校正的函数。不过更重要的是，MATLAB 提供了一个方便的作图和数值计算环境，在这个环境下可以非常轻松地将校正的指标和结果以图形的形式表示出来，结论一目了然。无论是理论推导还是工程应用，都有很强的辅助作用。

4.1 控制系统校正指标和经验公式

对于一个稳定的控制系统的要求通常可以归结为要求其输出量尽可能与输入量保持某种给定的关系(包括与输入量相等这种最简单的关系)。但是任何一个实际的控制系统都不能完全做到这一点,总存在着一定的误差。造成系统误差的原因很多,其中由于系统本身的结构和参数造成的误差称为原理性误差。此外,组成系统的元部件的不良特性,如摩擦、死区、间隙等非线性因素也都会产生误差。误差有静态的,也有动态的。从某种意义上说,误差的大小就可以表明一个控制系统性能的优劣。从这一点出发,工程上形成了一些指标,用以衡量控制系统的性能,这些性能指标尽管提法不同,但都体现了对系统静态特性和动态特性要求。

性能指标要反映系统实际性能的特点,又要便于量测和检验。所以对于不同类型的系统,对于不同的研究和应用领域,采用不同的性能指标。性能指标的提法有很多种,大体上可以归纳为两类:时间域指标和频率域指标。这两类指标在前面各章中都曾经简单介绍过,现在以图 4-2 MATLAB 绘制的二阶系统响应曲线为例再做一简单回顾。

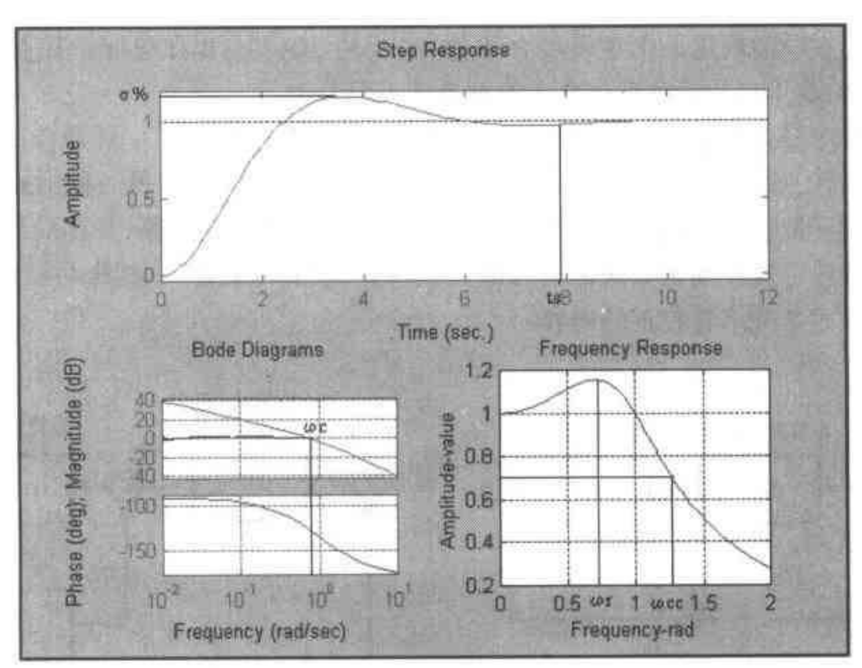


图 4-2 控制系统的性能指标

其中上半部分是系统的阶跃响应,横坐标是时间,纵坐标是响应值;左下部分是系统的 Bode 图,横坐标为频率,单位 rad/s,纵坐标分别是增益 dB 值和相角(°);右下部分是未取对数的系统频率响应曲线。

时间域指标包括静态指标与动态指标。

(1) 静态指标:指的是静态误差 e、无静差度、以及开环比例系数。这里 e 是指系统在跟踪典型输入(单位阶跃输入,单位斜坡输入和抛物线输入)时的静态误差。关于干扰和负载的变化所造成的静态误差也可以规定类似的指标。

(2) 动态指标：主要指的是过渡过程时间 ts 和超调量 σ%(见图 4-2 上半部分系统的阶跃响应)。此外，上升时间、延迟时间、峰值时间、振荡次数等也都属于时间域动态指标。但为了简单起见，通常只用过渡过程时间和超调量来表示系统的动态特性。

频率域指标包括开环频率指标和闭环频率指标。

(1) 开环频率指标。指的是截止角频率 ω_c，相角稳定裕量 γ、增益稳定裕量 Kg 等等。其中比较常用的是前两项。

(2) 闭环频率指标。主要指闭环谐振峰值 Mr 和谐振角频率 ω_r、闭环截止角频率 ω_{cc}(相当于闭环增益为 0.707(-3dB)时的角频率)。

无论是时间域指标还是频率域指标，都在一定程度上反映了系统的稳定性、快速性和准确性，只是反映的角度有所不同，所以各种指标之间必定存在着一定的联系。

不过还应看到，性能指标只是从某些工程角度大体上描述系统性能，有可能数学性质差别很大的系统，如不同阶次的系统，具有几乎相同的性能指标。因此不能指望在不同指标提法之间建立准确的数据对应关系。

但是在工程应用中，人们还是总结了一些经验公式来定性或定量的描述这些性能指标之间的关系。这些经验公式一般都是在时间中被证明是行之有效的，我们后边应用 MATLAB 对控制系统进行校正时也是以这些经验公式为基础的。不过如前所述，这些公式仅仅是经验性的，不具有同样的意义。

二阶系统的性能指标有着确定的关系，这里就不讨论了。对于高阶系统来说，在初步设计时可假设其谐振峰值 M_r 与相角裕量 γ、闭环截止角频率与开环截止角频率之间有如下的简洁关系：

$$M_r \approx \frac{1}{\sin\gamma}, \qquad \omega_{cc} \approx \omega_c$$

应当指出，不同性能指标并不是完全等价的，M_r 和 γ 虽然都在一定程度上反映了系统的稳定性，但 M_r 包含的信息要比 γ 多。γ 只反映某一特定频率下系统的性质，而 Mr 则反映了一定频率范围内系统的性质。M_r 值大的系统，一般可以说振荡剧烈(稳定性差)，但 γ 值大的系统，却不能保证一定有足够的稳定性。为了保证系统有足够的稳定裕量，一般希望 $M_r < 1.4$，$\gamma > 30°$，ω_c 的选择由系统的快速性要求来确定。以上述两式为基础，我们可以推导出一系列控制系统校正的经验公式：

1. $M_r \approx \frac{1}{\sin\gamma}, \sigma(\%) = \frac{2000}{\gamma} - 20$

2. $\sigma(\%) = \begin{cases} 100(M_r - 1), M_r \leqslant 1.25 \\ 50\sqrt{M_r - 1}, M_r < 1.25 \end{cases} \quad \sigma = 0.16 + 0.4(M_r - 1)$

$t_s = \frac{\pi}{\omega_c}[2 + (1.5M_r - 1) + 2.5(M_r - 1)^2]$

$M_r = \frac{h+2}{h-1}, 1.1 \leqslant M_r \leqslant 1.8,$

3. $t_s \approx (4 \sim 9)\omega_c$

其中 h 表示对数幅频特性图中频段斜率为 -1 的那一段宽度。为了简便起见，有时也把 h 称为中频段宽度。

4.2 系统开环频率特性设计

有了 MATLAB 这个得力的工具,系统的开环频率设计变得非常简便易行。我们只需选用几个简单的经验公式,不断地用 MATLAB 绘制的图形来校正控制器的设计,不必进行大量的计算,就可以得到满意的结果。本章的图形比较多,因为不是具体讲述图形绘制,而且到空间有限,所以尽量把曲线压缩在比较小的范围内,以能看清刻度和关键值为宜。本章实例的讲述格式也与前几章有所不同,以边求解边显示结果为主,主要是为了显示解题的思路。并且,考虑到读者对 MATLAB 已经比较熟悉,有关建立 M 文件和存盘等步骤就不再详细叙述了。请看下例:

[例 4.1] 考虑如图 4-3 所示的电机定位系统,其系统传递函数如下页所示。相关参数为:

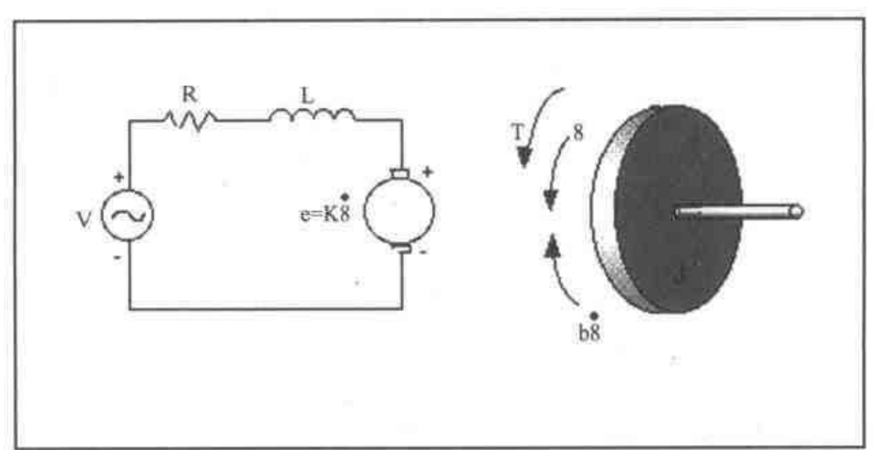


图 4-3 电机定位系统示意图

电机转动惯量 J = 3.2284E-6 kg.m²/s²,系统机械阻尼系数 b = 3.5077E-6 Nms,机电常量 $K = K_e = K_t$ = 0.0274 Nm/A,电阻 R = 4 Ω,电感 L = 2.75E-6 H。输入量是电源电压 V,输出量是转轴的角度 θ,并假设系统的转轴是刚性的。试求解一校正装置,使系统的阶跃响应满足:

$$\frac{\theta(s)}{V(s)}=\frac{K}{s((Js+b)(Ls+R)+K^2)}$$

(1) 过渡过程时间小于 40ms。

(2) 超调量小于 16%。

(3) 静态误差为零。

(4) 扰动静态误差为零。

结果:求得的校正装置为:

$$G(s)=\frac{0.0182(s+60)(s+52.9)}{s}$$

校正后系统的阶跃响应如图 4-4 所示。其横坐标是时间,纵坐标是系统的阶跃响应值。

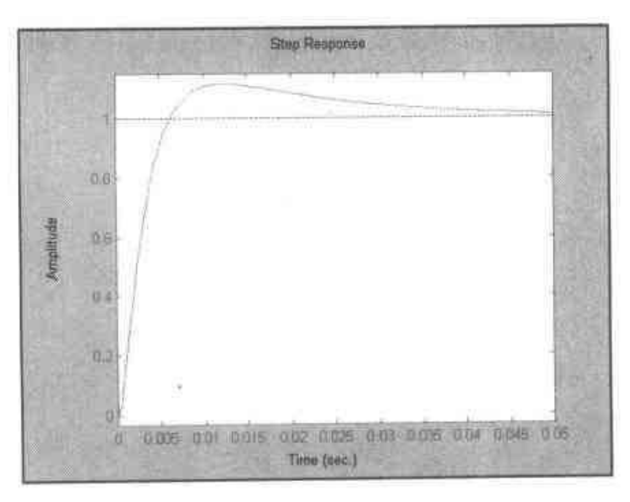


图 4-4 校正后电机定位系统的阶跃响应曲线

分析:校正装置是积分器加一个二阶的导前环节,功能是消除稳态误差,增强系统的快速性。从图 4-4 来看,系统的过渡过程时间小于 40ms,超调量小于 20%,并且稳态无差。基本满足要求。

求解过程:

本例的解题步骤分为以下几部分:

1. 首先绘制原系统的 Bode 图和阶跃响应、频率响应(见下页图 4-5)。

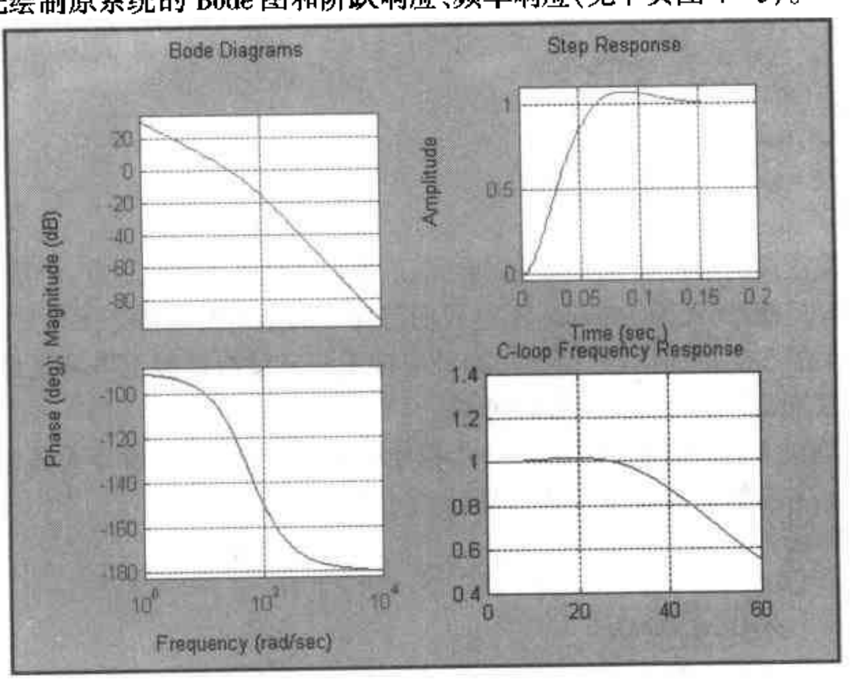


图 4-5 校正前系统的响应曲线

根据这些曲线分析与期望闭环系统的差异,大致确定校正装置的类型。在 MATLAB 环境下执行以下代码(可以在 MATLAB Command Window 下直接执行,但建议存为 M 文件,以便修改。以下同):

如图 4-5 所示左半部分是系统的 Bode 图,横坐标为频率,单位 rad/s,纵坐标分别是

增益 dB 值和相角(°);右边上半部分是系统的阶跃响应曲线,下半部分是未取对数的闭环系统频率响应曲线。

```
J = 3.2284E - 6;
b = 3.5077E - 6;
K = 0.0274;
R = 4;
L = 2.75E - 6;
num = [0 0 0 K];
den = [(J * L) ((J * R) + (L * b)) ((b * R) + K^2) 0];
numc = num;
denc = den + num;
w = logspace(0,4,101);
wt = 0:0.01:60;
t = 0:0.001:0.2;
g = freqs(numc,denc,wt);
mag = abs(g);
subplot(121),
bode(num,den,w);
subplot(222),
step(numc,denc,t);
grid;
subplot(224),
plot(wt,mag);
title('C - loop Frequency Response');
grid;
```

从系统校正前的阶跃响应曲线可以看到,虽然超调量基本满足要求,但系统的响应速度太慢,过渡过程时间约为 150ms。不过我们首先要满足对扰动静态无差的要求。因为如果最后解决静态误差的话,开始添加的频率校正环节就有可能不再满足设计要求了。这也是频域系统设计中一定要注意的问题。

2. 在系统的开环传递函数中添加一个积分器 1/s,满足系统对静态无差的要求

紧接着上一步的窗口或 M 文件,键入以下代码:

```
numi = 1;
deni = [1 0];
numiol = conv(num,numi);
deniol = conv(den,deni);
bode(numiol,deniol,w)
```

得到初步校正后系统的 Bode 图,如图 4 - 6 所示。其中上半部分是系统的相频特性曲线,下半部分是系统的幅频特性曲线。

3. 求解期望开环频率响应的截止角频率和曲线特性

设计要求超调量小于 16%,过渡过程时间小于 40ms,根据 4.1 节提供的经验公式,在 MATLAB Command Window 下输入下列代码:

```
ts = 0.04;
sigmac = 16;
wc = 9/ts;
gama = 2000/(sigmac + 20);
```

得到截止角频率和相角裕量为：

```
wc = 225
gama = 55.5556
```

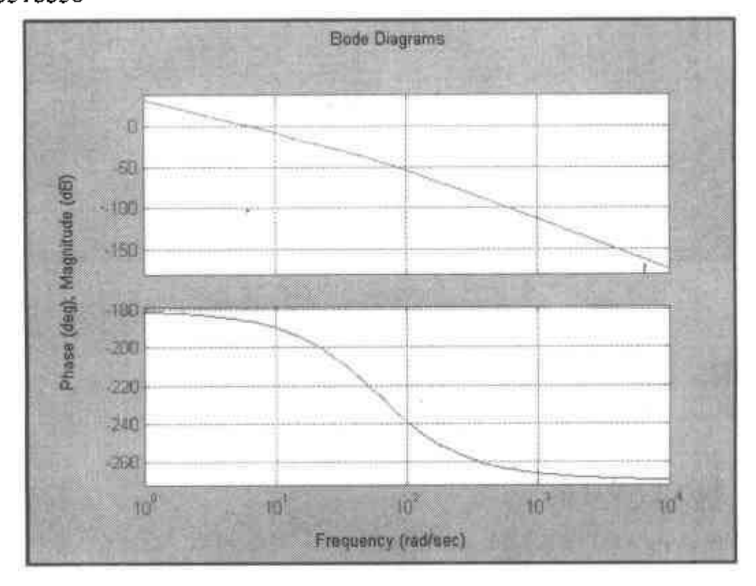


图 4-6　加入积分器后系统的 Bode 图

考虑到原系统阶次较高，取 $\omega_c = 250\text{rad/s}, \gamma = 50°$。对照图 4-6，发现 $\omega_c = 250\text{rad/s}$ 左右时系统的幅频特性约为 -60dB，相频特性约为 -260°。并且，从幅频特性曲线的斜率上可以看出，系统大约在 60rad/s 左右有一个极点，这个低频段与中频段结合部位的极点对系统的稳定性影响较大，应设法消去。

4．设计校正装置消去低频段与中频段结合部位的极点

准备采用积分加导前的校正装置 $\frac{s+60}{s}$，消去 60rad/sec 左右的极点。紧接上一步，输入如下代码：

```
numpi = [1 60];
denpi = [1 0];
%闭环传递函数
numpiol = conv(numpi, num);
denpiol = conv(denpi, den);
bode(numpiol, denpiol, w)
```

校正后系统的 Bode 图如图 4-7 所示。

从图 4-7 中可以看到，系统的幅频特性曲线是斜率约为 -2 的直线，而相频特性曲线在 180°左右变化。因此，对照 $\omega_c = 250\text{rad/s}$、$\gamma = 50°$ 的要求，需要在 $\omega_c = 250\text{rad/s}$ 左右补偿 50°的相角。

5．满足 $\omega_c = 250\text{rad/s}$、$\gamma = 50°$ 的要求

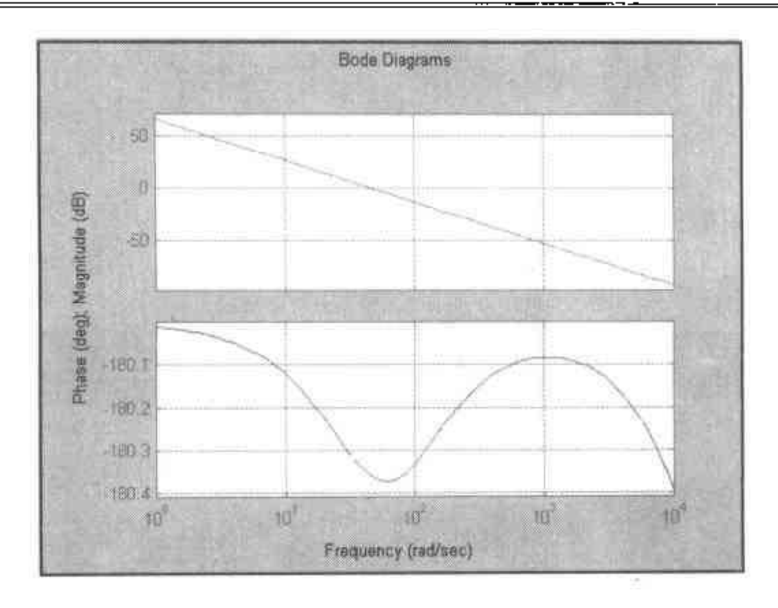


图 4-7 消去低频段极点后系统的 Bode 图

根据 4.1 节的经验公式，求得校正装置的传递函数为：

$$\frac{0.0189s+1}{5.8776e-4s+1}$$

紧接上一步，键入以下代码：

```
a=(1 - sin(gama * pi/180))/(1 + sin(gama * pi/180));
T=1/(wc * sqrt(a));
numpil = conv([1 60],[T 1]);
denpil = conv([1 0],[a * T 1]);
numpilol = conv(numpil,num);
denpilol = conv(denpil,den);
w = logspace(2,3,101);
bode(numpilol,denpilol,w)
```

校正后系统的 Bode 图如图 4-8 所示。

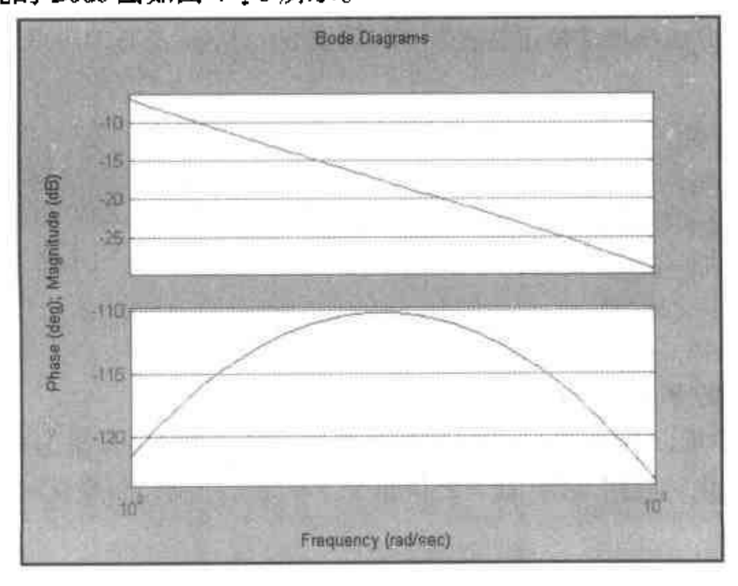


图 4-8 校正相角后的系统 Bode 图

从图 4－8 中可以看到，系统在 $\omega_c=250\mathrm{rad/s}$ 时相角不低于 $-120°$，满足了初步设计要求。但此时的幅频特性约为 －20dB，与期望值相差 －20dB。不过这一点改善起来比较容易，只需增大系统的低频增益即可。

6. 改善系统的低频增益并初步检验校正结果

与 －20dB 相对应的系统增益为 K＝10，因此，将系统的开环传递函数乘以 10，并绘制其变化阶跃响应曲线以检验初步设计的结果。紧接上一步，键入以下代码：

```
kpid = 10;
bode(kpid * numpilol,denpilol,w)
[numpilcl,denpilcl] = cloop(kpid * numpilol,denpilol,-1);
t = 0:0.001:0.1;
step(numpilcl,denpilcl)
```

此时系统的 Bode 图和阶跃响应图，如图 4－9 所示。

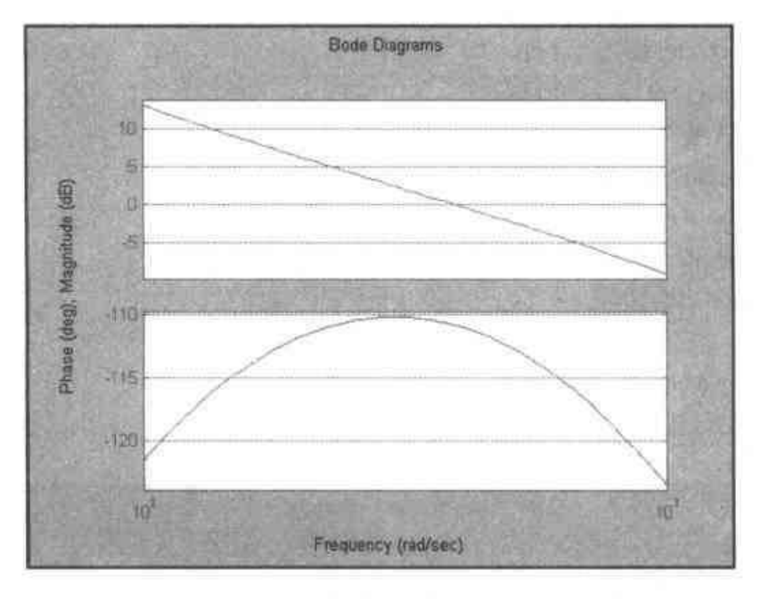


图 4－9　补偿增益后的系统 Bode 图

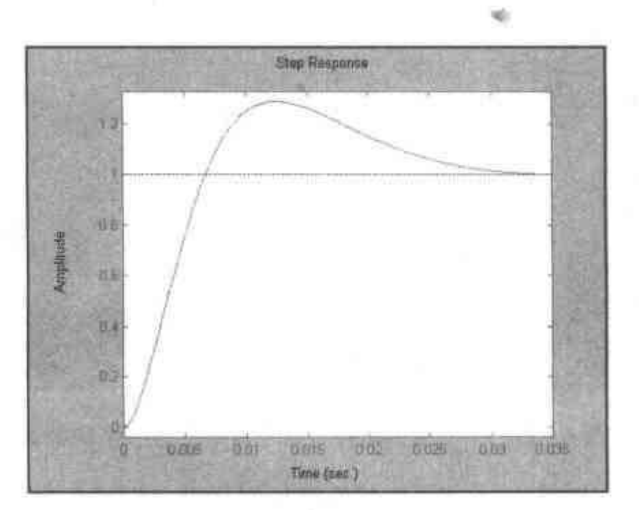


图 4－10　初步设计后的系统变化阶跃响应

其中上半部分是系统的相频特性曲线，下半部分是系统的幅频特性曲线。

从图 4－10 细心的初步设计结果中可以看到，此时系统的过渡过程时间约为 30ms，好于预期的 40ms；但超调量太大，约为 28％，比预期的 16％多了近 3/4，这在工程上是无法接受的。因此，还需要进一步设计，目标是减小系统的超调量。

7. 根据 4.1 节的经验公式，将系统的相角裕量扩大到 70°，并响应的在截止频率方面作一定补偿，令 $\omega_c = 300\text{rad/s}$。紧接上一步，键入以下代码：

```
%相角裕量
gama = 70;
%截止角频率
wc = 300;
%超前校正
a = (1 - sin(gama * pi/180))/(1 + sin(gama * pi/180));
T = 1/(wc * sqrt(a));
numpil = conv([1 60],[T 1]);
denpil = conv([1 0],[a * T 1]);
%闭环传递函数
numpilol = conv(numpil,num);
denpilol = conv(denpil,den);
%绘制 Bode 图
w = logspace(2,3,101);
kpid = 5;
bode(kpid * numpilol,denpilol,w)
```

此时系统的 Bode 图，如图 4－11 所示。

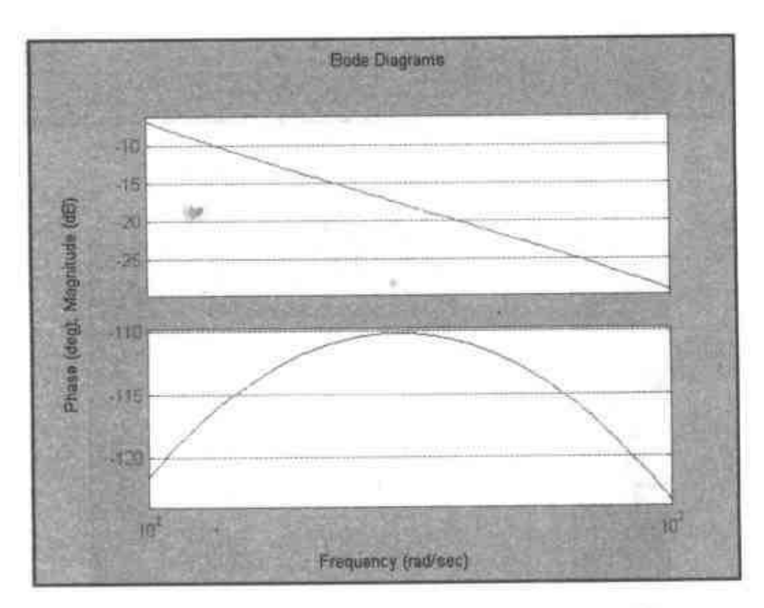


图 4－11 增大相角裕量和截止频率后的系统 Bode 图

从图 4－11 来看，与图 4－8 相同，都是相角裕量满足要求但低频增益过低，与期望值相差约 －18dB。

8. 补偿增益并最终检验校正结果

在 MATLAB Command Window 下计算求得:

```
10^0.9
ans = 7.9433
```

因此,选择增益 K = 8,输入以下代码:

```
kpid = 8;
bode(kpid * numpilol,denpilol,w);
%闭环系统传递函数
[numpilcl,denpilcl] = cloop(kpid * numpilol,denpilol, - 1);
t = 0:0.001:0.1;
%绘制阶跃响应曲线
step(numpilcl,denpilcl)
```

系统最终的 Bode 图如图 4 – 12 所示。

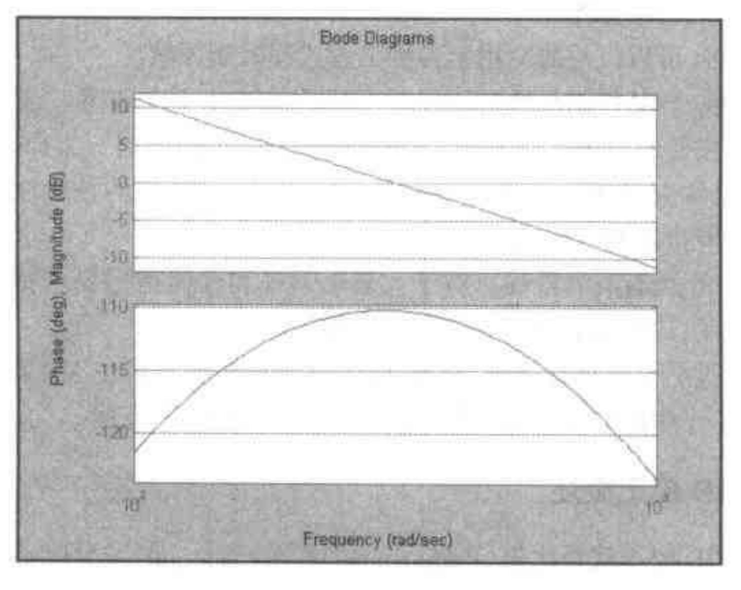


图 4 – 12 电机定位系统校正最终结果

系统最终的阶跃响应和校正装置曲线请参阅图 4 – 4 及结果,各项指标均满足系统最初的设计要求。

小结:从本例的解题过程可以看出,MATLAB 强大的绘图功能非常适合用来进行控制系统的频率域校正。在求解校正装置时可以逐步试探,所需要的性能指标从图中基本都可以读出,或者只需要根据经验公式进行简单的计算。而手工计算时不可能绘制出如此精确的图形,只能靠数值计算,这样作一方面计算量比较大,另一方面也不够直观,有可能出现较大的偏差。当然,使用 MATLAB 进行频率域控制系统设计也不会一帆风顺,本例的求解过程就大致分为初步设计和二次设计两部分。不过 MATLAB 的优点在于可以应用以前的结果迅速更改设计参数,不必重新进行繁琐的环境初始化和数据输入。因此 MATLAB 在控制领域被称为是“草稿纸式的演算语言”,就是赞扬它这种继承性的开放式结构。本例还用到了一个新的 MATLAB 函数 cloop(),其功能是根据系统的开环传递函数和反馈类型求解系统的闭环传递函数。通过上边的代码读者基本就可以理解其调用格

式和参数设置,详细情况请读者参阅 MATLAB 帮助。

4.3 串联校正

串联校正的主要内容是 PID 校正,即比例积分微分校正。事实上,在工程领域中应用简单串联校正的场合,80%都是 PID 校正。因此,本节就以 PID 校正为例,详细阐述串联控制的原理和应用。

在生产过程系统控制的发展历程中,PID 校正是历史最悠久、生命力最强的基本控制方式。在 20 世纪 40 年代以前,除在最简单的情况下可以采用开关控制外,它是惟一的控制方式。此后,随着科学技术的发展特别是数字式计算机的诞生和发展,涌现出许多新的控制方式。然而直到现在,PID 校正由于它自身的优点仍然是得到最广泛应用的基本控制方式。简单说来,PID 校正具有以下优点:

(1) 原理简单,使用方便。

(2) 适应性强,可以广泛的应用于各种工业过程控制领域。

(3) 鲁棒性强,即其控制品质对被控对象特性的变化不大敏感。这也是 PID 校正获得广泛应用的最主要的原因。一方面,它成本低廉,易于操作;另一方面,对于绝大部分控制对象,可以不必深究其模型机理,直接应用 PID 校正,其较强的鲁棒性保证了加入校正装置的系统的性能指标基本能满足要求。

当然,PID 校正也有其局限性。对于大延迟系统和性能指标要求特别高的系统 PID 校正就无能为力了,这就需要考虑更先进的控制方法。

本节将先通过一个简单的例子,介绍 PID 校正的原理及其在 MATLAB 下的实现方法,然后再根据一个串联校正实例,详细介绍 PID 校正的工程应用。

4.3.1 PID 校正概述

如前所述,PID 校正就是比例(Proportional)积分(Integral)微分(Derivative)校正的简称。传统的 PID 调节器的动作规律是:

$$u = K_p e + K_I \int e dt + K_D \frac{de}{dt}$$

$$K_p + \frac{K_I}{s} + K_D s = \frac{K_D s^2 + K_p s + K_I}{s}$$

这是典型的按偏差控制的负反馈结构。其中 e 是偏差,即输出量与设定值之间的差;u 是控制量,作用于被控对象并引起输出量的变化。Kp 是比例增益系数,其控制效果是减小响应曲线的上升时间及静态误差,但无法做到消除静态误差,因此,单纯的 P 校正是有差调节,一般不会单独使用。Ki 是积分增益系数,其控制效果是消除静态误差,I 校正是无差调节;但它会延长过渡过程时间,增大超调量,甚至影响系统的稳定性,因此,一般也不会单独使用。Kd 是积分增益系数,其控制效果是增强系统的稳定性,减小过渡过程时间,降低超调量。Kp、Ki、Kd 与系统时间域性能指标之间的关系间下表 4-1

表 4-1 PID 调节参数与系统时间域性能指标间关系

参数名称	上升时间	超调量	过渡过程时间	静态误差
Kp	减小	增大	微小变化	减小
Ki	减小	增大	增大	消除
Kd	微小变化	减小	减小	微小变化

上表的意义是 PID 参数增大时各系统性能指标的变化情况。当然，各参数与性能指标之间的关系不是绝对的，只是表示一定范围内的相对关系。因为各参数之间还有相互影响，一个参数变了，另外两个参数的控制效果也会改变。因此，在设计和整定 PID 参数时，上表只起一个定性的辅助作用。下面以表 4-1 为基础，希望通过这个例子，使读者了解 PID 校正的基本功能在 MATLAB 下实现的方法。由于空间有限，一些 PID 参数整定过程的图表就略去了，不过思路是相同的，读者可以自己掌握。

[例 4.2] 考虑图 4-13 我们熟悉的弹簧—阻尼系统，传递函数 G(s)如下，参数为 M = 1kg，b = 10 N.s/m，k = 20 N/m，F(s) = 1。试设计不同的 P、PD、PI、PID 校正装置，使相同的响应曲线满足：

(1) 较快的上升时间和过渡过程时间。

(2) 较小的超调量。

(3) 静态误差为零。

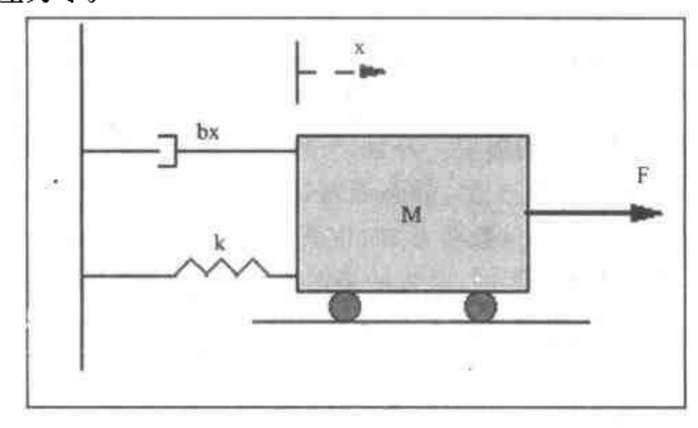


图 4-13 弹簧—阻尼相同示意图

结果：求得的校正装置为：

$$G(s) = \frac{X(s)}{F(s)} = \frac{1}{M_s^2 + bs + k}$$

P 校正：Kp = 300；

PD 校正：Kp = 300，Kd = 10；

PI 校正：Kp = 30，Ki = 70；

PID 校正：Kp = 350，Ki = 300，Kd = 50。

求解过程：

本例的解题步骤分为以下几部分：

1. 求解未加入校正装置的系统开环阶跃响应。

根据系统的开环传递函数,在 MATLAB Editor/Debugger 下编辑下述代码:

```
clear;
num = 1;
den = [1 10 20];
step(num,den)
```

得到系统的开环阶跃响应,如图 4-14 所示。

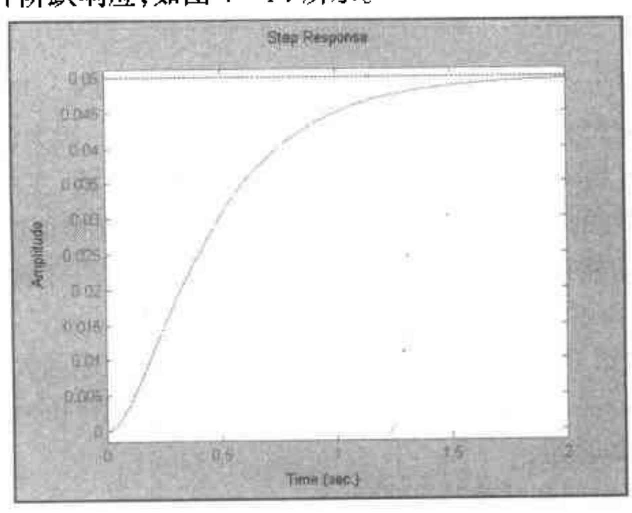


图 4-14 未加入校正装置时系统的开环阶跃响应

从图 4-14 中可以看出,系统的开环响应曲线未产生振荡,属于过阻尼性质。这类曲线一般响应速度都比较慢。果然,从图中读出,系统的上升时间约为 1 秒左右,过渡过程时间约为 1.5s 左右。对于一个虎克常数为 20N/M 的弹簧系统来说,这样的响应速度确实太满了,很难满足迅速定位的需要。并且,另外一个关键性的问题是,系统在幅值为 1 的阶跃响应输入下稳态值仅为 0.05,静态误差达 0.95,远不能满足跟随设定值的要求。这是因为系统传递函数分母的常数项为 20,也就是说系统对直流分量的增益是 1/20。因此,时间趋于无穷,角频率趋于零时,系统的稳态值就趋于 1/20 = 0.05。为了大幅度降低系统的静态误差,我们首先想到 P 校正。

2.P 校正装置设计

从表 4-1 中我们看到增大 Kp 可以降低静态误差、减小上升时间和过渡过程时间,因此首先选择 P 校正,也就是在系统中加入一个比例放大器。此时系统的闭环传递函数为:

$$G_c(s) = \frac{K_p}{s^2 + 10s + (20 + K_p)}$$

此时系统的静态误差为 $K_p/(20 + K_p)$。本着尽量减小静态误差的原则,我们取系统的比例增益 $K_p = 300$,这样可以把系统的静态误差降低到 0.06 左右。但并不是说系统的比例增益 K_p 可以无限制的越大越好,而是要受到实际系统的物理条件和放大器的实际情况限制,一般也就是取到几十或几百的量级。另外,增大系统的比例增益还可以改善系统的快速性。紧接上一步,输入以下代码:

```
Kp = 300;
num = [Kp];
%特征方程
```

```
den = [1 10 20 + Kp];
t = 0:0.01:2;
%系统阶跃响应
step(num,den,t)
```

得到加入 P 校正后系统的闭环阶跃响应如图 4－15 所示。

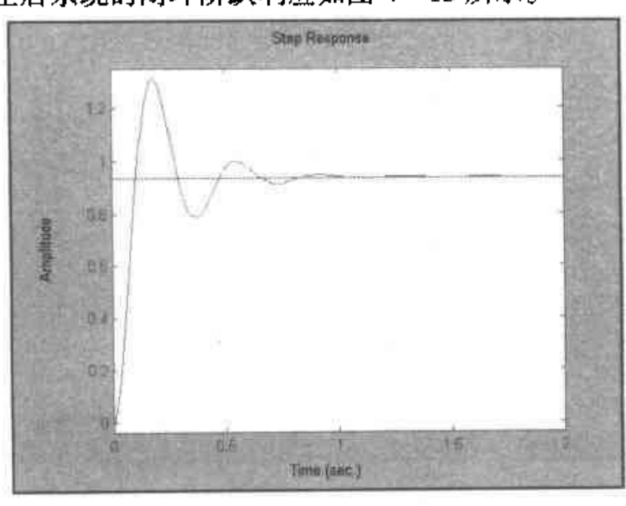


图 4－15 P 校正后系统的闭环阶跃响应曲线

从图 4－15 可以看到，系统的稳态值在 0.94 左右，静态误差约为 0.06，这和最初的设计要求是一致的。并且，曲线的形状从过阻尼型变为衰减振荡型，衰减比（响应曲线第一个峰值和第二个峰值之比）大于 4:1。因此，系统的快速性应该有一定的改善。从图中读出，系统的上升时间不超过 0.2s，过渡过程时间不超过 0.7s，验证了表 4－1 有关参数变化对系统性能影响的说法。

不过曲线由过阻尼该为衰减振荡又产生了另外一个问题，就是系统的超调量达到 30% 以上，第一个峰值振荡幅度过大。应该寻找另外的校正方式解决这个问题，下面我们尝试使用 PD 校正。

3. PD 校正装置设计

从表 4－1 中我们看到增大 Kd 可以降低超调量、减小过渡过程时间，对上升时间和静态误差影响不大。因此可以选择 PD 校正，也就是在系统中加入一个比例放大器和一个微分器。此时系统的闭环传递函数为：

$$G_c(s) = \frac{K_D s + K_p}{s^2 + (10 + K_D)s + (20 + K_p)}$$

仍选择 Kp = 300；Kd 一般选择为系统振荡周期倒数的 8 倍左右，也可以根据系统性能指标用 MATLAB 不断绘图校正。这里略去绘图选择的过程，给出 Kd = 10。代码如下：

```
Kp = 300;
Kd = 10;
%校正装置
num = [Kd Kp];
%特征方程
```

```
den = [1 10 + Kd 20 + Kp];
t = 0:0.01:2;
step(num,den,t)
```

得到加入 PD 校正后系统的闭环响应如图 4 – 16 所示。

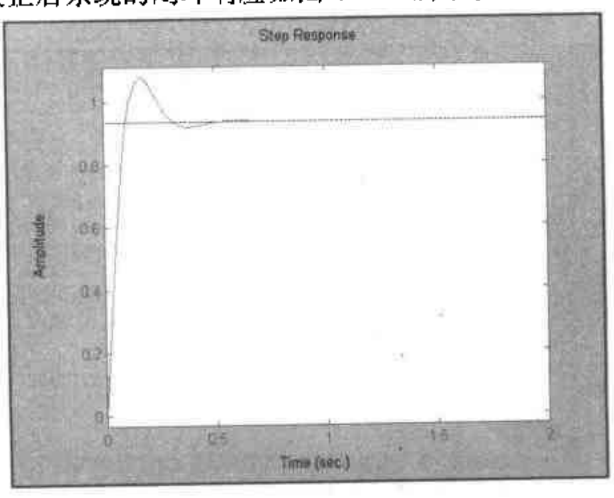


图 4 – 16　PD 校正后系统的闭环阶跃响应曲线

从图 4 – 16 可以看出，加入微分器后系统的响应曲线形状仍是衰减振荡型，但振荡次数显著减少，并且超调量也降低了不少。这就进一步验证了表 4 – 1 中有关增大 Kd 可以增强系统稳定性的说法。从系统的上升时间和静态误差来看，Kd 的变化对其影响不大。现在的问题就是静态误差不为零，这需要 PI 校正来解决。

4. PI 校正装置设计

从表 4 – 1 中我们看到增大 Ki 可以消除静态误差，因此可以选择 PI 校正，也就是在系统中加入一个比例放大器和一个积分器。此时系统的闭环传递函数为：

$$G_c(s) = \frac{K_p s + K_I}{s^3 + 10s^2 + (20 + K_p)_s + K_I}$$

考虑到 Ki 的作用，可以大幅度降低 Kp，取 Kp = 30。Ki 可以选得大一些，逐步降低。这里取 Ki = 70。输入代码：

```
t = 0:0.01:2;
step(num,den,t)
Kp = 30;
Ki = 70;
num = [Kp Ki];
den = [1 10 20 + Kp Ki];
%绘制阶跃响应曲线
t = 0:0.01:2;
step(num,den,t)
```

得到加入 PI 校正装置后系统的响应曲线如图 4 – 17 所示。

从图 4 – 17 中可以看到，加入 PI 校正后系统的稳态值为 1，即静态误差为零，系统的输出量最终能够无差的跟踪设定值的变化。不过由此又产生了另一个问题，系统的过渡

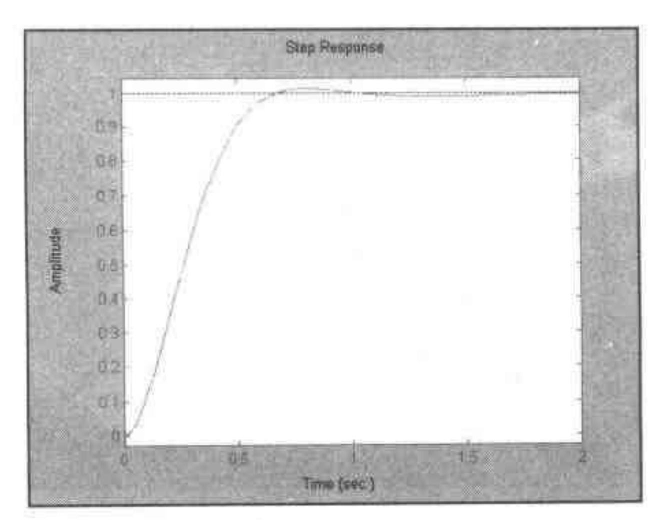


图 4-17 PI 校正后系统的闭环阶跃响应曲线

过程时间显著增加，快速性降低。这也是使用积分器带来的副作用。在这种情况下，如果希望系统各方面的性能指标都达到一个满意的程度，一般都要采用典型的 PID 校正。

5.PID 校正装置设计。

对于本例这种机械控制系统，采用 PID 校正一般都能取得满意的控制结果。其校正装置为：

$$G_c(s)=\frac{K_D s^2+K_p s+K_I}{s^3+(10+K_D)s^2+(20+K_p)s+K_I}$$

Kp、Ki 和 Kd 的选择一般是先根据经验公式确定一个大致的范围，然后通过 MATLAB 绘制的图形逐步校正。这里由于不考虑校正装置的可实现性，参数可以取的大一些。取 Kp = 350，Ki = 300，Kd = 50。输入代码：

```
Kp = 350;
Ki = 300;
Kd = 50;
%PID 校正装置
num = [Kd Kp Ki];
den = [1 10 + Kd 20 + Kp Ki];
%绘制阶跃响应曲线
t = 0:0.01:2;
step(num,den,t)
```

得到加入 PID 校正装置后系统的响应曲线如图 4-18 所示。

从图 4-18 可以看到，系统的响应曲线几乎已经和阶跃函数本身差不多了。当然，这只是一种理想状态的控制情况。并且还由于本例是机械系统，不存在延迟，而且快速性较好的缘故，实际上是很难达到的。

不过我们也应该看到，对于一般的控制系统来说，应用 PID 控制还是比较有效的，而且基本不用分析被控对象的机理，只根据 Kp、Ki 和 Kd 的参数特性以及 MATLAB 绘制的阶跃响应曲线进行设计即可。

小结：一般说来，在工程控制领域常用的是 PD 校正、PI 校正和 PID 校正。如果对控

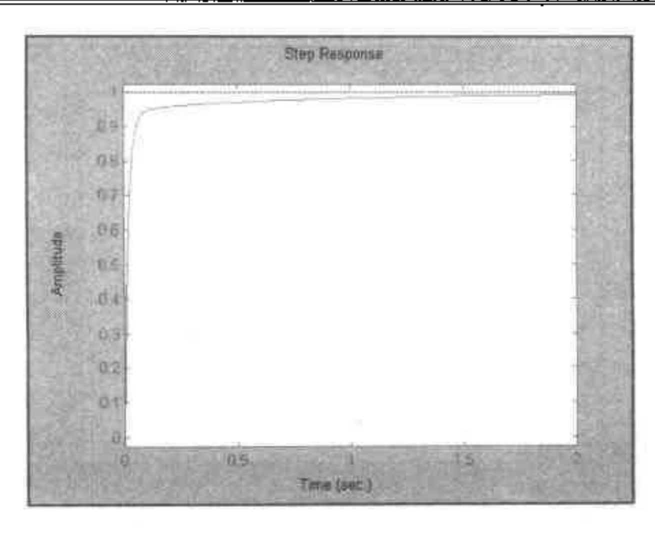


图 4-18 PID 校正后系统的闭环阶跃响应曲线

制品质要求不高,有时也会用到 P 校正。关于 PID 的参数整定,人们总结了许多经验图表和公式,比较著名的有动态特性参数法、稳定边界法、衰减曲线法以及 Ziegler-Nichols 经验公式等等。

但是在 MATLAB 环境下,我们甚至可以不借助这些经验公式,直接根据仿真曲线来选择 PID 参数。根据系统的性能指标和一些基本的整定参数的经验,选择不同的 PID 参数进行仿真,最终确定满意的参数。这样做一方面比较直观,另一方面计算量也比较小,并且便于调整。当然,在大多数情况下,根据经验公式进行计算还是有必要的,可以为我们指明参数选择的方向。

4.3.2 串联校正举例

上一小节讲述的 PID 校正主要是应用在时间域,相应的频率域提法分别叫做超前校正(PD 校正)、滞后校正(PI 校正)和滞后超前校正(PID 校正)。

超前校正的校正装置传递函数为:

$$G(s)=\frac{aTs+1}{Ts+1},a>1$$

其功能是在指定的频率附近产生相位超前角,提高系统的相角裕量。如果希望校正后的系统截止角频率为 ω,并在 ω 附近相对于原系统提供 θ 度的超前角,则:

$$\omega=\frac{1}{\sqrt{a}T} \qquad a=\frac{1+\sin\theta}{1-\sin\theta}$$

滞后校正的校正装置传递函数为:

$$G(s)=\frac{Ts+1}{\beta Ts+1},\beta>1$$

其功能是只降低中频段和高频段的开环增益而不影响低频段。如果希望校正后的系统截止角频率为 ω,并在 ω 附近原系统的增益为 L,则:

$$\frac{1}{T}=(\frac{1}{4}\sim\frac{1}{10})\omega \qquad L=20\lg\beta$$

滞后超前校正的校正装置传递函数为:

$$G(s)=\frac{T_2s+1}{\beta T_2s+1}\frac{aT_1s+1}{T_1s+1},\alpha,\beta>1$$

其功能是增加相位稳定裕量,改善系统动态性能的同时降低中频段增益,改善系统的静态性能。一般在单独使用超前校正或滞后校正无法实现控制目标的情况下采用滞后超前校正。请看下面这个例子:

[例 4.3] 某控制系统固有部分传递函数 G(s)如下,试分别设计串联校正装置 K(s),满足下列要求:

(1) 要求开环本例系数 $K\geq100$,相角裕量 $\gamma\geq30°$,截止角频率 $\omega\geq45$rad/sec;

(2) 要求开环本例系数 $K\geq100$,相角裕量 $\gamma\geq40°$,截止角频率 $\omega=5$rad/sec;

(3) 要求开环本例系数 $K\geq100$,相角裕量 $\gamma\geq40°$,截止角频率 $\omega\geq20$rad/sec;

$$G(s)=\frac{1}{s(0.1s+1)(0.01s+1)}$$

结果:求得的校正装置分别为:

(1) 超前校正:$K(s)=100\frac{0.063s+1}{0.0063s+1}$

(2) 滞后校正:$K(s)=100\frac{0.8s+1}{14.22s+1}$

(3) 滞后超前校正:$K(s)=100\frac{0.25+1}{1.253s+1}\frac{0.137s+1}{0.0182s+1}$

分析:因为给出的要求都是频率域的指标,所以用超前或滞后的串联校正比较合适。当然,用 4.2 节的开环频率设计法也可以。相角裕量是保证系统稳定性的,设计过程中一般要留出 5°~12°的滞后量;截止角频率是控制系统通频带的指标,设计完之后一般要用 4.1 节的经验公式校验一下系统的阶跃响应。

求解过程:

本例的解题步骤分为以下几部分:

1. 求解原系统的响应曲线,对照性能指标作出分析。

在 MATLAB Editor/Debugger 下编辑以下代码:

```
%清除内存变量
clear;
num = 1;
den1 = [1 0];
den2 = [0.1,1];
den3 = [0.01,1];
den = conv(den1,conv(den2,den3));
%系统闭环传递函数
[numc,denc] = cloop(num,den, - 1);
t = 0:0.01:1.5;
wt = 0:0.5:60;
%系统频率响应数据
g = freqs(numc,denc,wt);
mag = abs(g);
%绘制图形
```

```
subplot(121),
bode(num,den);
subplot(222),
step(numc,denc,t);
subplot(224),
plot(wt,mag);
title('Frequency Response - Amplitude');
xlabel('Frequency - rad');
ylabel('Amplitude');
grid;
```

得到校正前系统的响应曲线如图 4－19 所示。

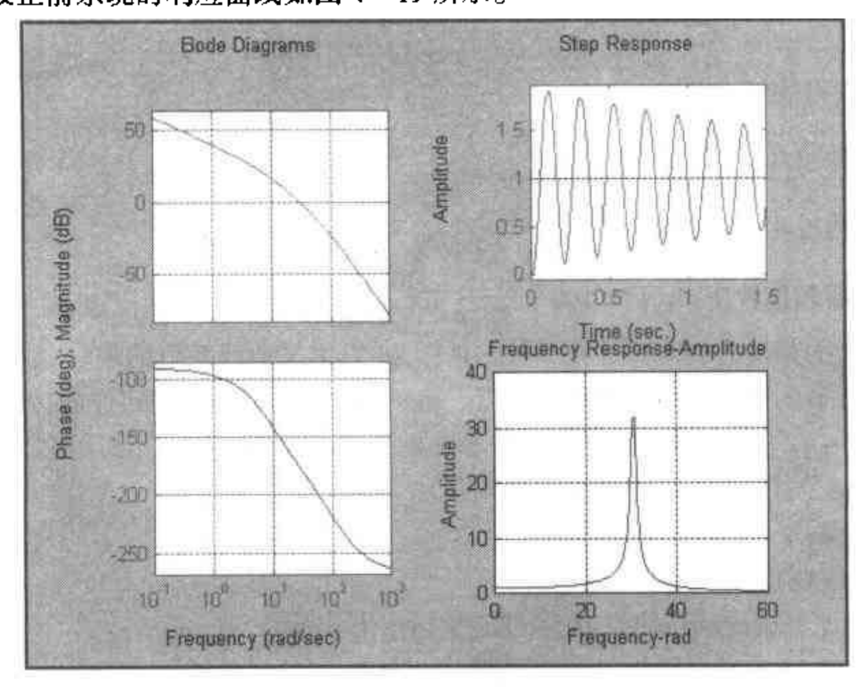


图 4－19 校正前系统的响应曲线

其中左半部分是系统的 Bode 图，横坐标为频率，单位 rad/s，纵坐标分别是增益 dB 值和相角(°)；右边上半部分是系统的阶跃响应曲线，下半部分是未取对数的闭环系统频率响应曲线。

从图 4－19 中可以看到，最明显的是系统的变化阶跃响应曲线振荡幅度非常剧烈，并且衰减频率很慢，系统濒临不稳定。对应系统的频率响应曲线上在 $\omega = 30\text{rad/s}$ 处有一个非常高的谐振峰，约为 32 左右。谐振峰高说明系统对该频率的输入容易引起共振，形成衰减很小的振荡曲线。而阶跃函数是包含一切频率分量的，因此对照阶跃响应曲线可以看出系统的振荡频率约为 $2\pi/32 = 0.2$ 秒左右。这样的系统是无法投入使用的，因此必须进行串联校正。

2. 满足目标 1——超前校正。

根据校正的目标，对照系统的 Bode 图可以发现：对于目标 1，$\omega = 45\text{rad/s}$ 时系统的相角已经接近 $-190°$，并且增益的 dB 值为负。因此应该在 $\omega = 45\text{rad/s}$ 处补充 50°的相角超

前量，使其此处增益的 dB 值接近于零。所以应采用超前校正。当然，考虑到对系统开环增益的要求，还应该将系统的增益值 K 乘以 100。紧接上一步，输入以下代码：

```
%相角裕量
gama=55;
%截止角频率
wc=50;
%超前校正装置
a=(1+sin(gama*pi/180))/(1-sin(gama*pi/180))
T=1/(wc*sqrt(a))
num=[0 0 0 100];
numa=[a*T 1]
dena=[T 1]
numao=conv(num,numa);
denao=conv(den,dena);
bode(numao,denao);
```

得到系统经超前校正后的 Bode 图如图 4-20 所示。

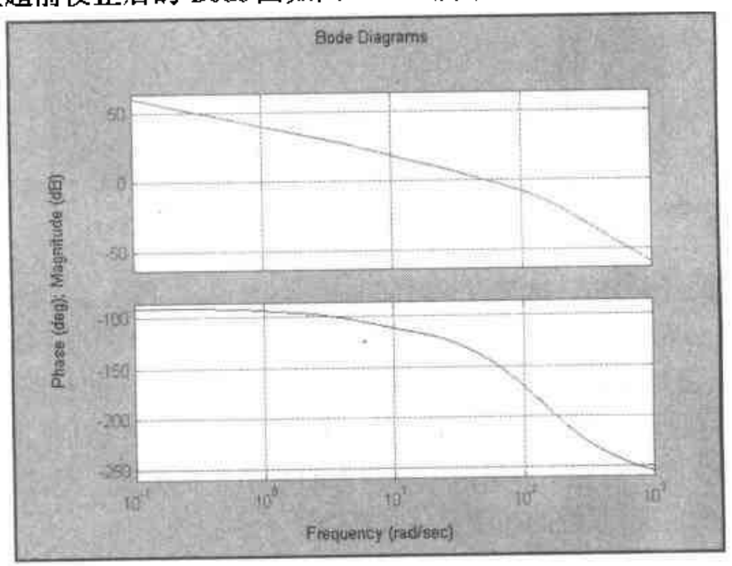


图 4-20　加入超前校正后系统的 Bode 图

其中上半部分是系统的相频特性曲线，下半部分是系统的幅频特性曲线。

对照图 4-20 和图 4-19，可以看到，系统的截止角频率从 $\omega=20$rad/s 后移到了约为 $\omega=45$rad/s 处，对应的相角约为 $-140°$，相角裕量 40°，满足相角裕量 $\gamma\geq30°$的要求。这些结果验证了超前校正可以在一定的频率区域内提供相角超前量的说法。从 MATLAB Command Window 下可以读到校正装置的参数：

```
a = 10.0590
T = 0.0063
%校正装置传递函数多项式
numa = 0.0634 1.0000
    dena = 0.0063 1.0000
```

numa 和 dena 就是求得的超前校正装置传递函数的分子和分母多项式系数。从上边的代码中还可以看到，这里选补偿的相角超前量 gama = 50(°)，暂定的截止角频率 $\omega = 50(\mathrm{rad/s})$。这是因为从图 4-19 的系统 Bode 图来看，$\omega = 45\mathrm{rad/s}$ 处的相角约为 $-190°$，要满足相角裕量 $\gamma \geq 30°$ 的要求必须提供大于 40° 的相角超前量。考虑到传递函数间的相互影响取相角超前量为 50°。同理，取 $\omega = 50\mathrm{rad/s}$。同时还可以输入以下代码作时间域的验证：

```
[numac,denac] = cloop(numao,denao,-1);
step(numac,denac);
```

得到系统的阶跃响应曲线如图 4-21 所示。

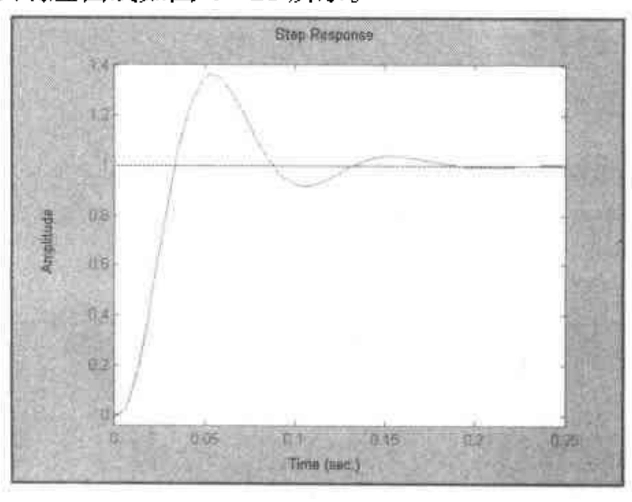


图 4-21　加入超前校正后系统的阶跃响应曲线

根据 4.1 节的经验公式，过渡过程时间 $ts = (4 \sim 9)/\omega$。代入 $\omega = 45\mathrm{rad/s}$ 后求得系统的过渡过程时间 ts 在 0.09s 至 0.2s 之间。对照图 4-21，可以读出系统的过渡过程时间约为 0.18s 左右，满足设计要求。不过从图中还可以看到，系统的超调量仍然比较大，大约在 35% 左右。前边说过，频率域的超前校正相当于时间域的 PD 校正。而从时间域性能指标上来说，PD 校正对完全抑制系统超调量的效果并不明显。不过，相对与未加校正时系统剧烈振荡的情况，系统的超调量还是降低了不少。

3. 满足目标 2——滞后校正

根据校正的目标，对照系统的 Bode 图可以发现：对于目标 2，$\omega = 5\mathrm{rad/s}$ 时系统的增益值约为 25dB，相角约为 $-120°$，超过了相角裕量要求，可以通过适当引入滞后相角。但中频段系统增益过高，对于这种系统，超前校正是无能为力的，应该采用降低中频段增益的滞后校正。紧接上一步，输入以下代码：

```
wc = 5;
g = 25;
%滞后校正装置
beta = 10^(g/20);
T = 4/wc;
wt = logspace(-2,2);
numb = [T 1];
denb = [beta * T 1];
```

```
numbo = conv(num,numb);
denbo = conv(den,denb);
bode(numbo,denbo,wt);
```

得到系统经滞后校正后的 Bode 图如图 4－22 所示。

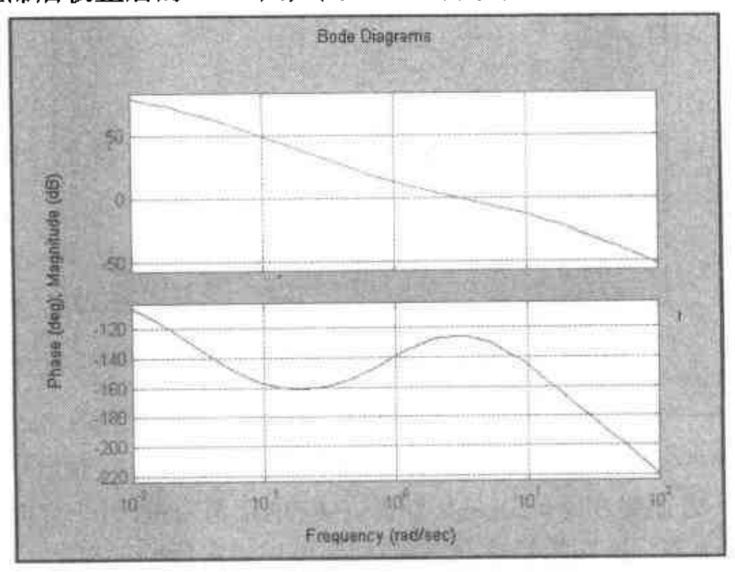


图 4－22　加入滞后校正后系统的 Bode 图

对照图 4－22 和图 4－19，可以看到，系统的截止角频率从 $\omega = 20$rad/s 前移到了约为 $\omega = 5$rad/s 处，对应的相角约为 $-130°$，相角裕量 $50°$，满足相角裕量 $\gamma \geq 40°$的要求。这些结果验证了滞后校正可以降低中频段和高频段系统增益的说法。当然，校正装置在低频段引入了一定的相角滞后量，不过因为系统的截止频率在中频段，在满足相角裕量要求的情况下增加一些相角滞后量对系统的稳定性并不会产生太大影响。从 MATLAB Command Window 下可以读到校正装置的参数：

```
beta = 17.7793
T = 0.8
%校正装置传递函数多项式
numb = 0.8 1.0000
denb = 14.2234 1.0000
```

numb 和 denb 就是求得的滞后校正装置传递函数的分子和分母多项式系数。从上边的代码中还可以看到，这里暂定的截止角频率 w = 5，T = 4/w；从图 4－19 中读出 $\omega = 5$rad/s 处系统的增益为 25dB。同时输入以下代码，可以求得加入滞后校正装置后系统的阶跃响应曲线：

```
[numbo,denbo] = cloop(numbo,denbo,1);
step(numbo,denbo);
```

此时系统的阶跃响应曲线如图 4－23 所示。

根据 4.1 节的经验公式，代入 $\omega = 5$rad/s 后求得系统的过渡过程时间 ts 在 0.8s 至 2s 之间。对照图 4－21，可以读出系统的过渡过程时间约为 1.8 秒左右，满足设计要求。如前所述，频率域的滞后校正相当于时间域 PI 校正，其效果是消除系统静态误差，降低中频段增益。

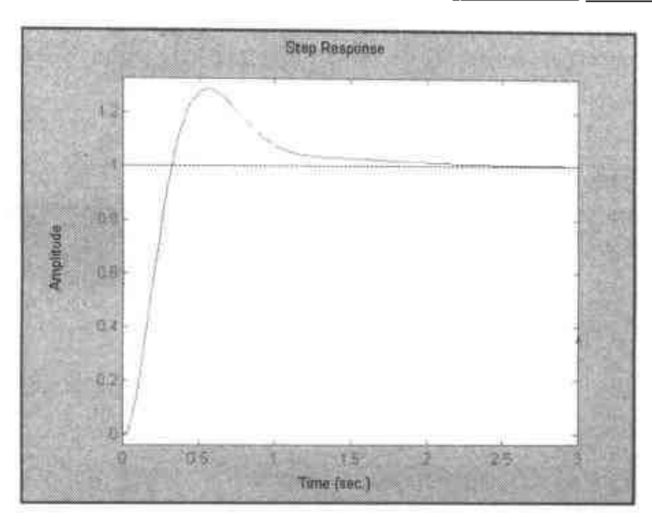


图 4-23　加入滞后校正后系统的阶跃响应曲线

满足目标 3——滞后超前校正。根据校正的目标，对照系统的 Bode 图可以发现：对于目标 3，ω = 20rad/s 时系统的增益值约为 0dB，相角约为 -180°，相角裕量几乎为零。要达到校正目标，一方面要在 ω = 20rad/s 处提供 40°的相角裕量，另一方面又必须保证此时的系统增益仍然为零。这样的要求，单独的超前校正或滞后校正都是无法满足的。因此，必须采用综合这两种校正优点的滞后超前校正。首先设计超前校正装置，紧接上一步，输入下列代码：

```
%相角裕量
gama = 50;
%截止角频率
wc = 20;
%校正装置
a = (1 + sin(gama * pi/180))/(1 - sin(gama * pi/180))
T1 = 1/(wc * sqrt(a))
%系统传递函数
numa = [a * T1 1]
dena = [T1 1]
numao = conv(numa,num);
denao = conv(dena,den);
bode(numao,denao,wt);
```

得到系统的 Bode 图如图 4-24 所示。

对照图 4-24 和图 4-19，可以看到，系统的截止角频率从 ω = 20rad/s 后移到了约为 ω = 80rad/s 处。ω = 20rad/s 处系统的增益值约为 15dB，系统的相角约为 -130°。如果能把系统的响应曲线在 ω = 20rad/s 处下移 15dB 同时不影响低频段增益值，就能同时满足截止角频率 ω = 20rad/s、相角裕量 γ≥40°和开环增益 K≥100 的要求。这正是滞后校正的功能，因此，继续设计一个滞后校正装置，将系统在 ω = 20rad/s 处的增益下移 15dB。紧接上一步，输入以下代码：

```
g = 14;
%滞后校正装置
```

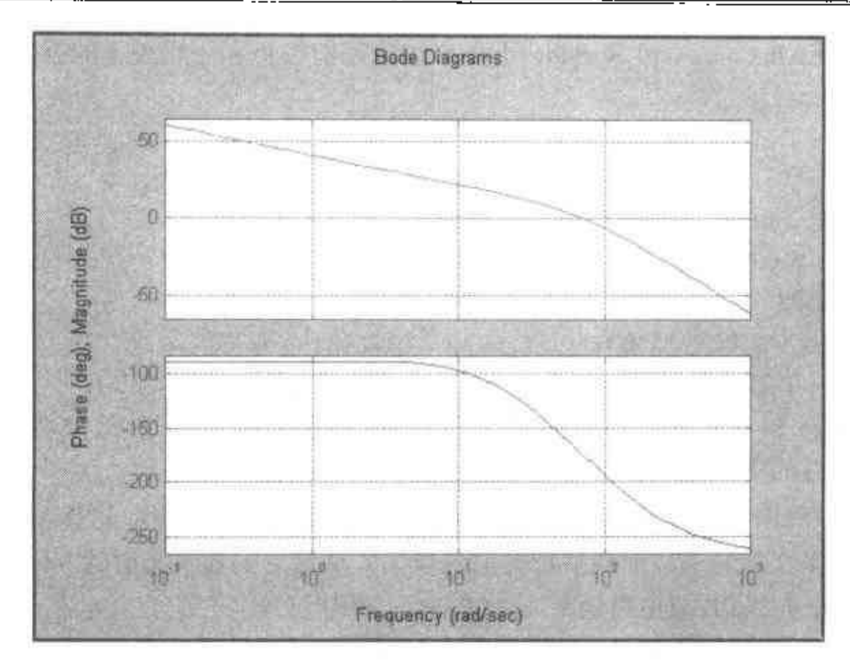


图 4-24 加入超前校正后系统的阶跃响应曲线

```
beta = 10^(g/20);
T = 5/wc;
wt = logspace( - 1,3);
%系统传递函数
numb = [T 1];
denb = [beta * T 1];
numo = conv(conv(num,numa),numb);
deno = conv(conv(den,dena),denb);
bode(numo,deno,wt);
```

得到加入滞后校正后系统的响应曲线如图 4-25 所示。

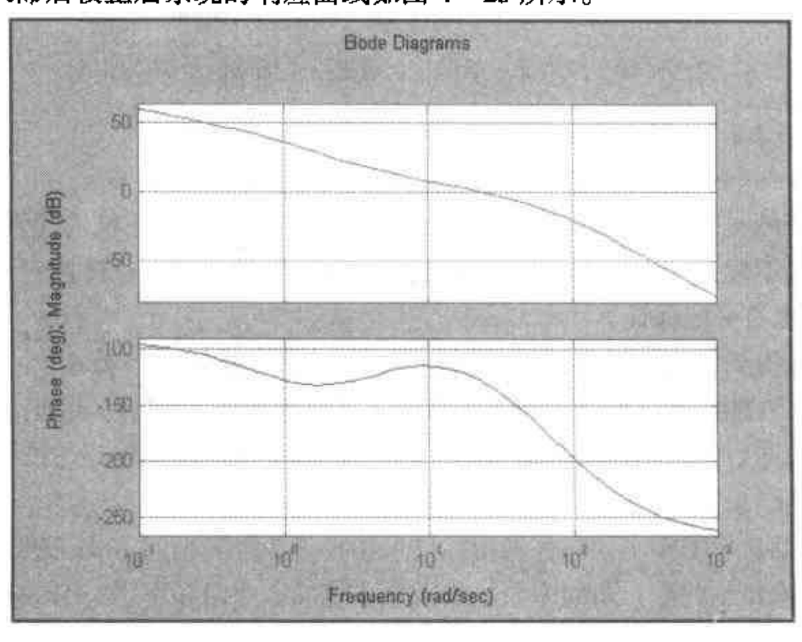


图 4-25 加入滞后超前校正装置后系统的 Bode 图

从 MATLAB Command Window 下可以读到超前校正和滞后校正的参数：

```
a = 7.5495
T = 0.0182
numa = 0.1374    1.0000
dena = 0.0182    1.0000
beta = 5.0000
T = 0.25
numb = 0.2500    1.0000
denb = 1.2500    1.0000
```

其中 numa 和 dena 是求得的超前校正装置传递函数的分子和分母多项式系数，numb 和 denb 是求得的滞后校正装置传递函数的分子和分母多项式系数。关于代码中各参数选择的理由和前面单独设计超前校正和滞后校正的理由基本一数，这里就不再赘述了。当然，有些是多次绘制响应曲线后选择的最佳值。从最后的效果来看，图 4-25 中系统的低频段增益超过 50dB，截止角频率 $\omega = 20$rad/s，对应相角 $-130°$。满足相角裕量 $\gamma \geq 40°$ 和开环增益 $K \geq 100$ 的要求。还可以得到此时系统的阶跃响应曲线如图 4-26 所示。

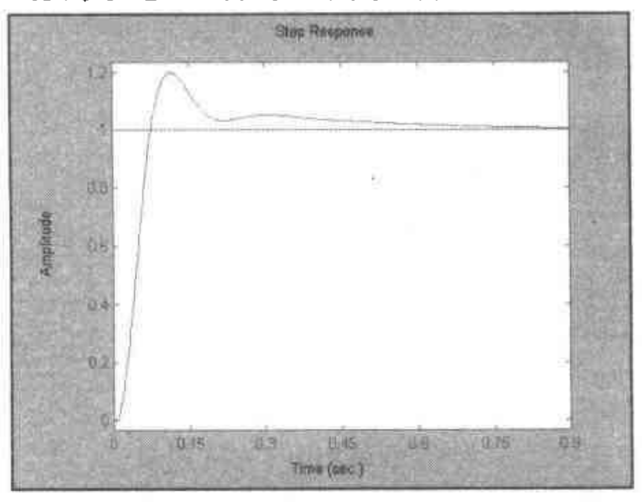


图 4-26　加入滞后超前校正装置后系统的阶跃响应曲线

```
[numc,denc] = cloop(numo,deno,-1);
step(numc,denc);
```

此时系统的时间域动态性能就比较令人满意了。一方面是系统的超调量约为 20%，并且振荡次数不超过 2 次；另一方面系统的上升时间约为 0.1s，过渡过程时间不超过 0.4s，快速性也有一定保证。

前边曾经提到，频率域的滞后超前校正相当于时间域的 PID 校正。一般来说，普通的动态或静态性能指标要求通过调节 PID 校正的参数都可以实现。因此，如果发现单纯的超前校正或滞后校正不能满足设计要求的话，可以尝试应用滞后超前校正，一般都能得到较为满意的结果。

小结：仔细比较本例不同的性能指标要求和校正装置实现的方式，我们可以发现下面这个规律：如果希望的截止角频率明显高于原系统的截止角频率（如 45rad/s 和 20rad/s），一般采用超前校正，在新的截止角频率处提供一定的相角超前量；如果希望的截止角频率

明显低于原系统的截止角频率(如 5rad/s 和 20rad/s),并且新的截止角频率处系统的增益值大于 0dB,则一般采用滞后校正,将新的截止角频率处增益下调为零;如果希望的截止角频率和原系统的截止角频率差不多,但相角裕量要求较高,一般采用滞后超前校正,在新的截止角频率处提供一定的相角超前量,并将其增益下调为零。

通过上面这个例子,我们可以发现使用 MATLAB 来实现这样的工作是非常方便的。并且,大量的图形使得设计者可以随时掌握系统的情况,修正其设计方案,这也算是一种解决问题的负反馈吧。

4.4　根轨迹校正

有关根轨迹的基本概念和绘制方法,在 3.3 节中已经有比较详细的论述。事实上,除了用来判断系统稳定性以及参数变化对系统性能的影响之外,根轨迹还可以作为校正控制系统的手段使用。

4.4.1　Rltool 环境概述

MATLAB 提供了一个辅助设计闭环系统根轨迹的仿真软件 Rltool,可以用来进行根轨迹校正。与著名的 Simulink 一样,Rltool 也采用了可视化的设计方法,在 MATLAB Command Window 下键入 rltool,即可进入 Rltool 的仿真界面,如图 4－27 所示。

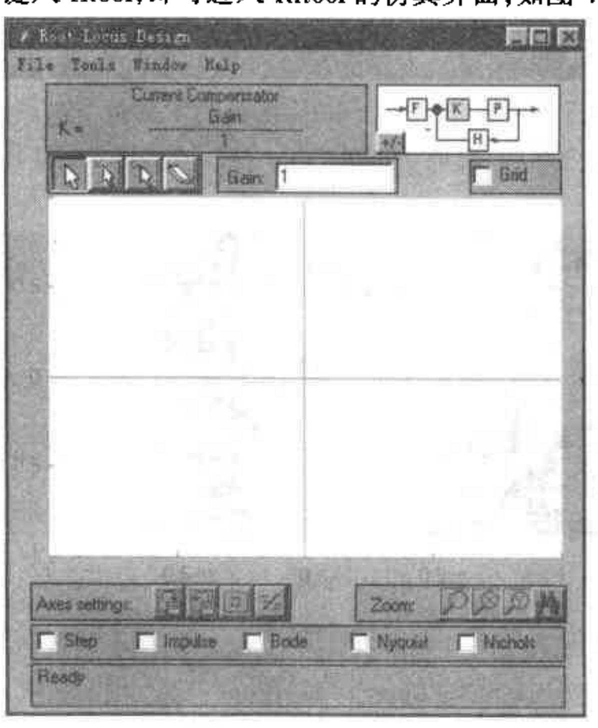


图 4－27　Rltool 仿真界面

在 Rltool 环境下只需设计很少的环境参数即可进行根轨迹的仿真和校正。并且,根轨迹校正装置设计完毕后还可以直接从图中读出,不需进行大量烦琐的计算。尤其是在不必精确计算模型参数的场合,只需在图中大致确定变化极点的位置及增益,即可获得校

正装置的模型，非常适合于工程设计。

选择图 4－27 中“File”菜单的“Import Model...”选项，即可进入如图 4－28 所示的 LTI 系统模型设计界面，输入系统的描述模型。图 4－28 右上角的方框图是系统的零极点传递函数描述模型，其中 F 和 H 是输入和反馈环节，没有特殊要求的话一般设为 1；K 是需求解的根轨迹校正装置，不在此界面输入；P 是被控对象的传递函数，本界面下可选中图 4－28 中间方框“Workspace Contents”下列表的系统描述，然后选择左边的箭头。

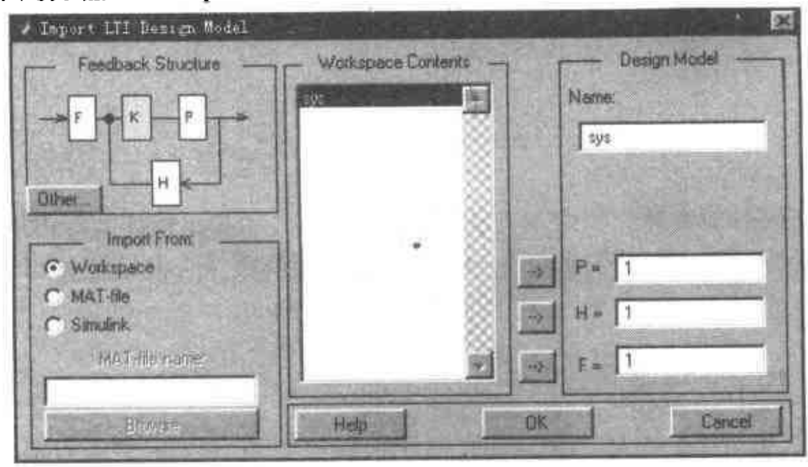


图 4－28　Rltool 环境下系统模型的输入

选择图 4－27 中“File”菜单的“Import Compensator...”选项，即可进入图 4－29 所示的 LTI 系统校正装置输入界面。该界面包括校正装置的名称“Name”对话框、零点选择复选框“Zeros”、极点选择复选框“Poles”以及相关的帮助信息。我们可以通过“Name”对话框赋予校正装置一个名称，选取下边的“Add Zero”按钮和“Add Pole”按钮为校正装置增加零极点。

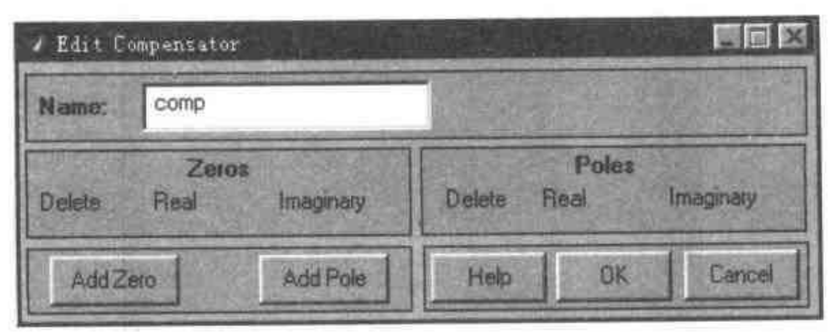


图 4－29　Rltool 环境下校正装置模型的输入

校正装置的输入方式与图 4－28 所示的 LTI 系统模型设计界面相同，但一般校正装置的传递函数都是先使用缺省的 Gain/1，在 Rltool 仿真界面下不断绘制其根轨迹图，通过根轨迹的分布情况和趋势逐步的调整确定最佳的参数。

当然，也可以在 MATLAB Command Window 下设计好初步的校正装置传递函数进行输入（其方法和以前介绍的生成系统描述的方法相同，直接输入传递函数分子和分母多项式即可），再逐步调整。

直接用鼠标点击图 4－27 左上角的系统方框图描述的对应环节，也可以输入相应的

传递函数。不过这就是手工编辑，而不是从“Workspace contents”中输入了。例如点击“K”环节，可以进入图 4－30 的校正装置编辑界面。

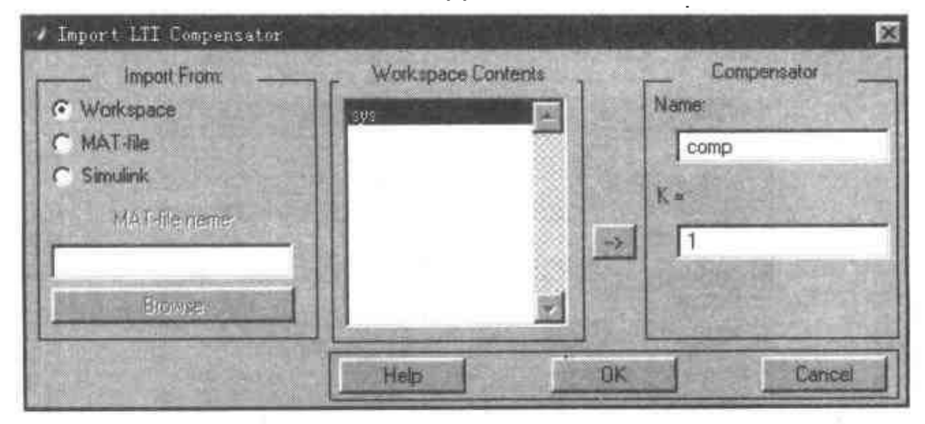


图 4－30　Rltool 校正装置编辑界面

该校正装置的模型是以零极点传递函数的形式描述的。修改“Name”选项可以更改校正装置的名称，对于大型的控制系统设计来说，这是十分必要的。选择“Add Zero”或“Add Pole”按钮可以增加校正装置的零点或极点，增加零点的界面如图 4－31 所示。

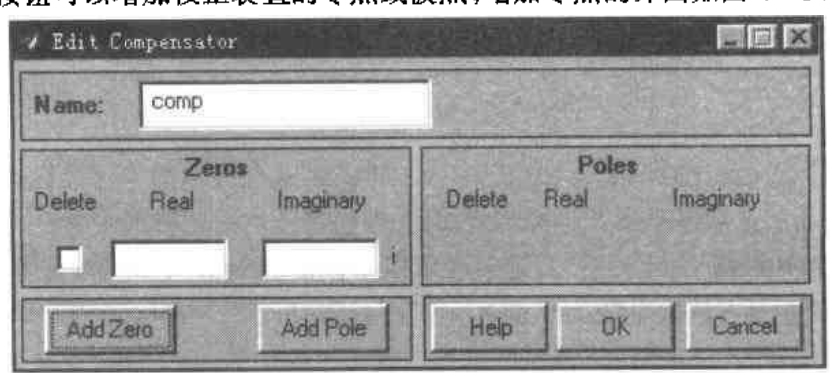


图 4－31　增加校正装置的零点

如果选中图 4－31 的“Delete”复选框，则表示删除该零点。增加极点的界面和选择方式与图 4－31 相同，这里就不再赘述了。

如果待校正的被控对象不是非常复杂，控制精度要求不高，可以考虑使用图 4－27 右上角第二行的四个按钮来设置系统的闭环极点和系统的开环零极点位置。

图 4－27 按下的缺省的按钮的功能是选择校正装置的增益。系统模型输入完毕画出闭环根轨迹后按下该按钮拖动鼠标在图 4－27 下方的坐标图中移动，图 4－27 第二行中间的“Gain”文本框的数值就会发生变化，找到合适的闭环极点(注意，是闭环极点，而非开环极点)位置后单击鼠标左键，即可完成闭环极点设置，同时“Gain”文本框中的数值就是校正装置的增益。

相应的，按钮和按钮的功能就是设置系统开环极点和零点位置的按钮，其使用方法与设置系统闭环极点的按钮相同。不过在实际设计根轨迹的工程中，一般是先确定校正后系统的零极点，然后在画好的根轨迹上确定闭环的极点位置，而不在根轨迹上的点是不能作为闭环极点的。

按钮的功能是取消设置完的系统开环零极点，只需选中该按钮后用鼠标在需要取消的位置上单击即可。

图 4－27 下方还有几个工具条，其中“Axes Settings”是用鼠标设置坐标轴，不过该选项的功能可由“Tool ”菜单下“Set Axes Preference”选项完成，如图 4－32 所示。

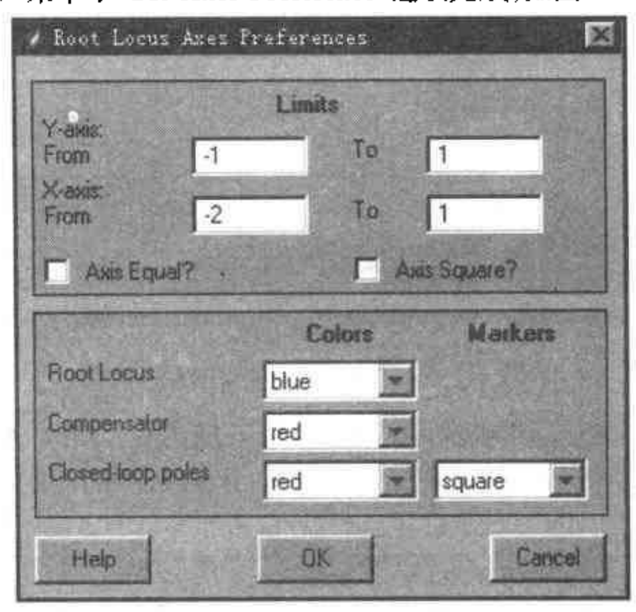


图 4－32　根轨迹坐标轴设置

各选项的意义都比较明显，只是其中“Axis Equal”选项表示横坐标和纵坐标的刻度相等，“Axis Square”选项表示不管坐标刻度如何，根轨迹图均为正方形。

图 4－27 中的　　　　　工具条表示以不同的比例和可视区域显示根轨迹坐标图。各选项从左至右依次是“Zoom in X－Y”、“zoom in X”、“zoom in Y”和“Full View”。其中按钮代表的“Full View”选项可以提供根轨迹的全景，便于设计者从总体上把握根轨迹的变化趋势。

图 4－27 还有一个工具条 Step　Impulse　Bode　Nyquist　Nichols ，其功能是分别绘制闭环系统的阶跃响应曲线(Step)、脉冲响应曲线(Impulse)、Bode 图(Bode)、Nyquist 图(Nyquist)和 Nichols 图(Nichols)。这个工具条深得控制系统分析及设计人员的喜爱。有了它，我们就不必将求得的校正装置和原系统模型重新在 MATLAB Command Window 下仿真，根据 plot()语句或 lsim()函数绘制这些图形，进行时间域或频率域指标的校验了。

4.4.2　根轨迹校正举例

按根轨迹校正反馈系统可以依以下步骤来进行：

(1) 根据给定的动态性能指标，确定希望的闭环极点的位置。若是二阶系统，直接求出其极点；高阶系统则求其主导根点。

(2) 绘制未校正系统的根轨迹。若希望的主导极点不在此根轨迹上，而且可以看出此根轨迹不可能提供满意的动态性能，那么就说明不能只靠改变增益 K 来满足系统的动态性能要求，必须使用校正装置来改造闭环系统的根轨迹，使其通过希望的主导极点。在

手工或 MATLAB Command Window 环境下，需要根据前边讨论的规则设计相应的校正装置，使其满足要求。而 Rltool 仿真环境提供了一些可视化操作手段，不必进行繁琐的计算，用鼠标即可将其配置到满意的位置

(3) 若校正后(或校正前的)根轨迹已经通过希望的闭环主导极点，则应检验相应的开环比例系数是否满足要求，如不满足，可采用在原点附近增加开环偶极子的办法来提高开环比例系数，同时保持根轨迹仍通过希望的主导极点。

(4) 校核系统的闭环静态和动态指标。

请看下面这个例子：

[例 4.4] 设某二阶被控系统的传递函数 G(s)如下。试选用合适的方法设计一个串联校正装置 K(s)，使得系统的阶跃响应曲线超调量 $\sigma\% < 30\%$，过渡过程时间 $ts < 1.5s$，开环比例系数 $K > 10/s$。

$$G(s) = \frac{K}{s(s+2)}$$

结果：初步求得的串联校正装置的传递函数为：

$$G(s) = \frac{49.6(s+3.9)}{(s+9.4)}$$

加入偶极子后系统的校正装置传递函数为：

$$G(s) = \frac{49.6(s+3.9)(s+0.12)}{(s+9.4)(s+0.02)}$$

分析：对于本例这种二阶系统，系统闭环阶跃响应的超调量 $\sigma\%$ 和过渡过程时间 ts 与其传递函数的系数有着确定的关系。即便加入校正装置后成为了三阶或四阶系统，其主导极点和超调量 $\sigma\%$ 和过渡过程时间 ts 之间同样有着近似的函数关系。因此，通过配置系统闭环极点的根轨迹校正的方法无疑是解决本例的最佳思路。另外，本例系统的阶次比较低，根轨迹的出发点、会合点和分支数都比较少，便于应用 MATLAB 提供的 Rltool 环境进行直观的设计和求解。

求解过程：

本例的求解分为以下几个部分：

1. 输入被控系统并绘制其根轨迹。

在 MATLAB Command Window 下键入以下代码：

```
%系统传递函数
num = 1;
den = [1 2 0];
%生成抽象的系统描述
sys = tf(num,den);
%调用 rltool 仿真环境
rltool(sys)
```

其中 num 和 den 分别是待校正的系统传递函数模型的分子和分母多项式系数。tf()是 MATLAB 提供的生产系统传递函数模型抽象描述的函数。键入回车，则进入如图 4-27 所示的 Rltool 根轨迹设计及仿真界面。选择该界面的“File”菜单的“Import Model...”选项，进入如图 4-33 所示被控对象模型输入界面。

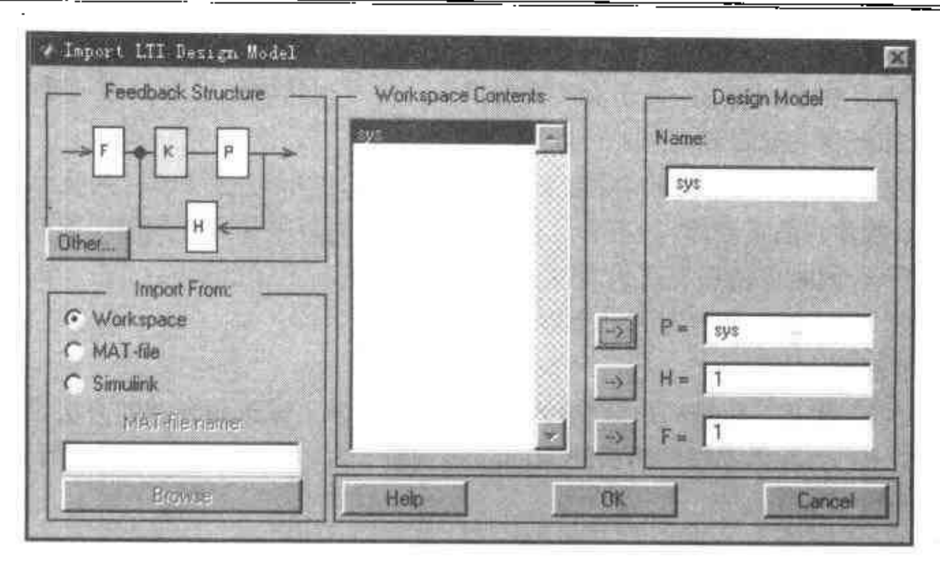


图 4－33 被控对象模型输入界面

选中图中“Workspace Contents”下列表的“sys”选项，然后单击“P”选项旁边的箭头，即可完成被控系统模型的输入。其功能是将刚才在 MATLAB Command Window 下编辑的系统 sys 的传递函数输入到环节“P”中。“P”下边的“H”和“F”分别表示反馈环节和输入环节的传递函数，本例另 H = 1，F = 1 表示单位负反馈，参考输入为 1。决定后选取“OK”按钮即可。

2. 根据系统根轨迹和阶跃响应评价其动态性能，并计算希望的闭环主导极点位置。

选取图 4－32 模型输入界面“OK”按钮后即返回如图 4－27 所示的 Rltool 根轨迹设计及仿真界面，此时以绘制完成系统的根轨迹，如图 4－34 所示。

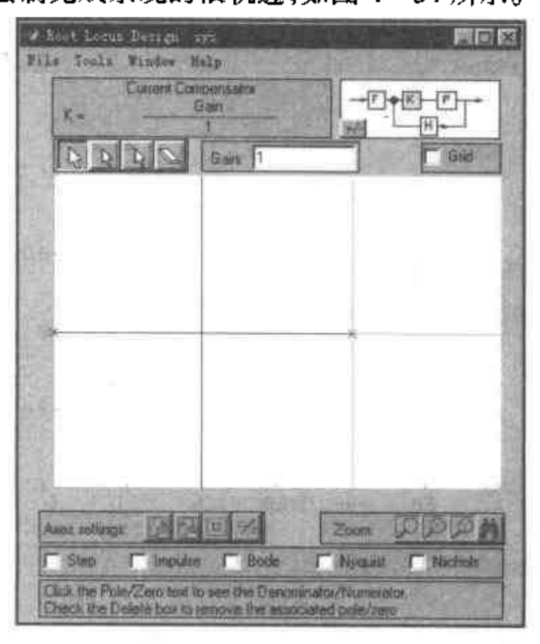


图 4－34 校正前系统的根轨迹

从图中可以看出，无论系统的闭环极点在其根轨迹上怎样变化，与原点的距离都比较接近。这样的系统过渡过程时间较长，很难满足 ts < 1.5s 的要求。选中图 4－34 的

"Step"选项,可得系统的阶跃响应,如图 4-35 所示。

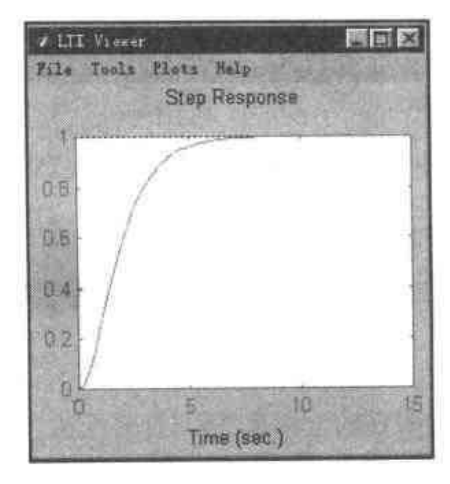


图 4-35 校正前系统的阶跃响应

果然,从图 4-35 中读出,系统的过渡过程时间超过 5s,并且曲线的形状是过阻尼型,快速性很差。

根据系统的动态指标超调量 $\sigma\% < 30\%$,过渡过程时间 $ts < 1.5s$,以及二阶系统极点与动态指标的关系:

$$\sigma = e^{-(\zeta/\sqrt{1-\zeta^2})\pi}$$

$$t_s = \frac{4T}{\zeta}$$

$$P_1\ P_2 = \frac{\zeta}{T} \pm j\frac{\sqrt{1-\zeta^2}}{T}$$

可以求得希望的系统闭环主导极点为:

$$P_1, P_2 = -3 \pm j3\sqrt{3} \approx -3 \pm j5.1962$$

数值计算都可以由相应的 MATLAB 数学运算函数实现,在第一章里已经有详细的介绍,这里就略去了。

3. 计算开环零极点位置并输入。

从图 4-34 可以看出,系统校正前的根轨迹不过希望的闭环极点,因此,必须设计一校正装置,将系统的根轨迹向左弯曲。一般校正装置的传递函数为:

$$K(s) = \frac{s-z}{s-p}$$

具体的设计方法有两种:一种是根据 3.3 节绘制根轨迹的模条件和角条件求出相应零点 z 和极点 p;另一种是根据一定的规则在 Rltool 环境下不断试验,直到选出满意的参数。对于本例来说,希望系统的根轨迹向左弯曲,一般是在负实轴上设置一个零点和一个极点,并且零点在极点的左边。弯曲程度随着零极点间距离的增大而增大(这些经验的具体论述在有关根轨迹的控制理论书籍中都可以找到,注意,是经验,而非严格的定理)。本例直接给出结果见图 4-36,零点 $z = -3.9$,极点 $p = -9.4$。考虑到物理可实现性,选择的位置都落在实轴上,虚部为零。

4. 坐标轴设置及校正后根轨迹绘制。

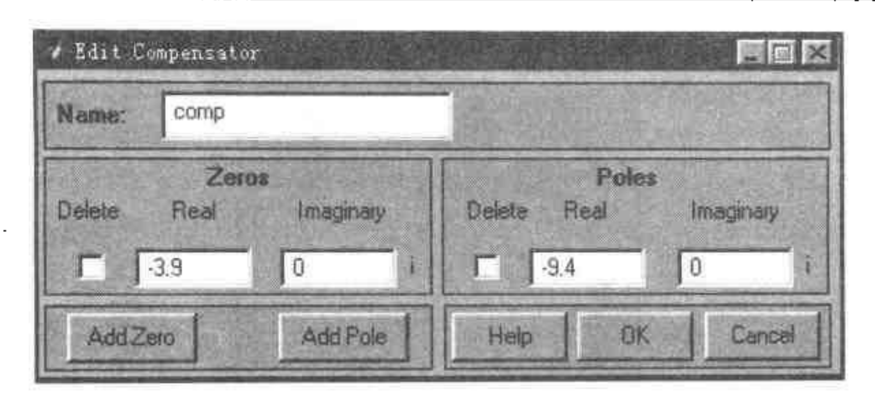


图 4－36　输入校正装置的零极点

从图 4－34 来看，根轨迹所在的坐标系范围过小，无法直接在原图的左边平面上选择校正装置零极点的位置。因此，首先要修改坐标范围，使之能够显示大范围的根轨迹图。

在图 4－33 界面下，选择“Tool ”菜单下“Set Axes Preference”选项，坐标系修改界面见图 4－37。

图 4－37　根轨迹坐标平面设置

将横坐标“X－axis”和纵坐标“Y－axis”的下限值均设为－10。由于 Rltool 仿真环境下坐标系的选择都是对称的，因此上限值可以不必手工设置，Rltool 仿真环境可以自动将其与下限值对应起来。当然，也可以只修改上限值，不修改下限值。因为根轨迹关于实轴对称，因此只设定坐标轴的下限或上限即可。至于中间的两个复选框“Axis Equal”和“Axis Square”，这是选择坐标系的外观的。如果用户希望 X 轴和 Y 轴采用同样的刻度，那么就选中“Axis Equal”；如果用户希望不管刻度如何，整个坐标系呈正方形，则选中“Axis Square”。我们这里使用缺省的“Axis Equal”选项。图 4－37 下边的部分是根轨迹颜色和标记的设置，一般选择缺省设置即可。

回到 Rltool 仿真界面后选择 Zoom 工具条的“Full View”按钮，即可见到类似下页图 4－38 的根轨迹图形。不同的是此时见到的图形没有图 4－38 的三个红色小方块和网格。这三个红色小方块是系统闭环极点所在的位置，其设置方法是：选择按钮，

在求得的主导极点之一(例如 $-3+j5.1962$)位置上单击鼠标左键,即可得到这三个红色小方块。同时"Gain"文本框的数值由 1 变为 49.6。当然,位置稍有偏差,"Gain"文本框的数值也稍有不同,不过基本上在 45～55 之间。至于网格,是因为选中了"Grid"复选框。该网格是按照极坐标绘制的,从中我们可以读出根轨迹相应点的模和幅角。至此,初步设计已经完成,下面是校验性能指标。

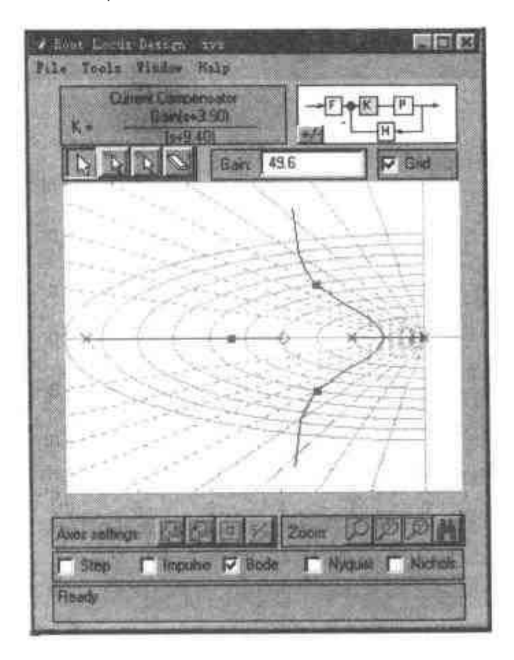


图 4－38　初步校正后系统的根轨迹图

5. 根据响应曲线校验系统的性能指标

选择图 4－38 的"Step"和"Bode"复选框,可以得到系统的响应曲线,如图 4－39 所示。

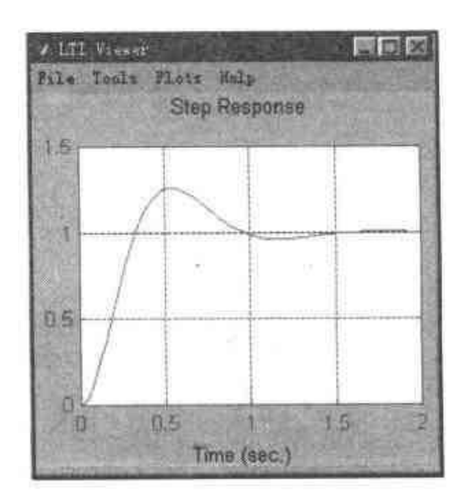


图 4－39　初步校正后系统的阶跃响应曲线

图 4－39 是此时系统的闭环阶跃响应曲线。从图 4－39 中可以读出,系统的超调量 $\sigma\%$ 约为 25%,过渡过程时间 ts 不超过 1.2s。满足超调量 $\sigma\% < 30\%$,过渡过程时间 $ts < 1.5s$ 的要求。因此,校正后动态指标是合格的。

系统的 Bode 图如图 4-40 所示，因为系统存在一个积分环节 1/s，所以其开环增益从第一个非零极点所在的位置 $\omega=2$ 处读取，约为 20dB，即 10/s 左右。本例要求是开环增益 K>10/s，因此，该校正装置在这个指标的表现上并不出色。如果投入实际应用，很可能造成不必要的麻烦。

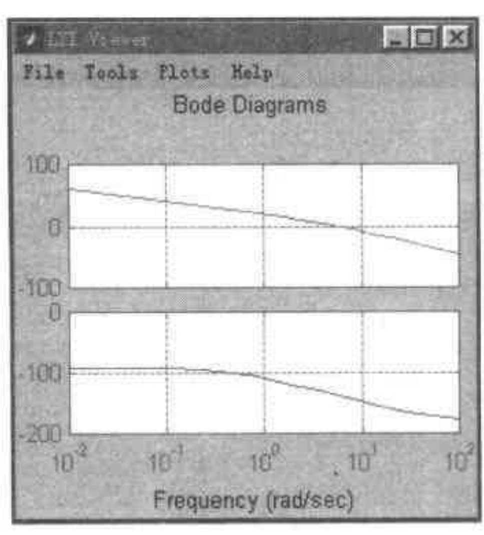


图 4-40 初步校正后系统的 Bode 图

为了解决这个问题，还应该在原来的基础上重新设计一个校正装置，使其满足静态开环增益 K>10/s 的要求。这时有的读者可能会说，把原校正装置的 K(s)增益"Gain"增大不就可以了吗。很可惜，这种简单的方法是不能成立的。从前边的分析我们也可以看出，改变增益"Gain"后系统闭环极点的位置势必也要发生变化，这样一来，系统的静态性能固然可以满足目标，动态性能就无法保证了。还得重新设计。能不能找到一种校正装置，即可以改变系统的静态开环增益，又不影响其动态性能？答案是肯定的，这就是下一步要用到的偶极子。

6. 增大开环增益——偶极子设计

所谓偶极子，是指复平面上距离很近的一对零点和极点。我们已经知道，有 n-r 个非零极点的系统的开环增益为：

$$K = K'\frac{\prod_{i=1}^{m}(-z_i)}{\prod_{j=1}^{n-r}p_j}$$

如果在原点附近加入一对偶极子 z 和 p，并且令 z=10p，系统的开环增益就变为：

$$K = K'\frac{\prod_{i=1}^{m}(-z_i)}{\prod_{j=1}^{n-r}p_j}\cdot\frac{(-z)}{(-p)} = 10K$$

因此，增加在原点附近的闭环偶极子可以改变系统的开环增益。并且，如果系统的主导极点远离这对偶极子，系统的根轨迹就不会发生太大变化。因为根轨迹开始于极点截止于零点，这对偶极子自己形成了一段根轨迹，不会影响到其它根轨迹。因此，系统的动态性能也不会因为这对偶极子而产生明显的变化。

这就回答了上一步提出的问题，我们找到了一种即可以改变系统的静态开环增益，又

不影响其动态性能的校正装置。考虑到系统的变化主导极点为 $-3\pm j5.1962$，模为 6。根据偶极子的设计原则，应该让偶极子中模较大的一个与系统主导极点的模相差 50 倍以上，并且使偶极子的零点和极点的模相差一定的倍数。因此，我们设计一对偶极子 -0.12 和 -0.02，将校正装置变为：

$$G(s)=\frac{49.6(s+3.9)(s+0.12)}{(s+9.4)(s+0.02)}$$

为了在 Rltool 仿真界面下配置这对偶极子，首先要点击 Zoom 工具条的“Zoom in X”按钮，将横坐标轴放大到可以辨认的程度，然后在分别用和按钮选择合适的位置放置该偶极子（当然可以从“File”菜单下的“Import Compensator”选项里输入，也可以用“Tool”菜单下的“Edit Compensator”选项设置），用删除。偶极子配置完成后，其结果如图 4－41 所示。

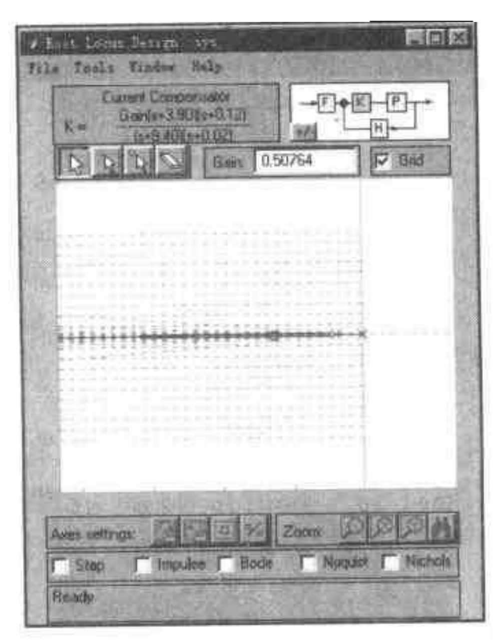


图 4－41　偶极子配置

从图 4－41 中可以看到，圆圈代表的零点在 -0.12 附近，叉号代表的极点在 -0.02 附近，它们之间有一条根轨迹，这也符合根轨迹起于极点终于零点的规律。此外，设置好零极点以后，上边校正装置 K 也变成了希望的形式。

从图 4－41 里还可以看到，系统根轨迹的横坐标变成了从 -0.2 到 0.05，而纵坐标不变。这样一来，配置该校正装置的偶极子和观察此范围内的根轨迹就变得比较方便了。这段根轨迹之间有一个红色小方块，这是一个闭环极点。不过它离主导极点很远，对系统的动态性能没有什么太大的影响。

如果希望看到此时根轨迹整体的情况和变化，可以点击按钮，得到此时系统的根轨迹的全景图如图 4－42 所示。

从图 4－42 中可以看出，系统的根轨迹的形状没有什么太大的变化，尤其是闭环主导极点所在的两条根轨迹仍然是向左弯曲的两条曲线。而闭环主导极点的位置也没有什么变化，依然在 $-3\pm j5.2$ 左右。

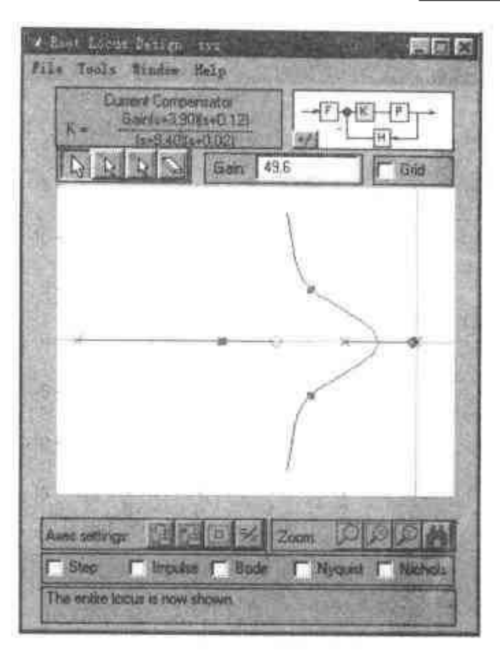


图 4－42 校正后系统的根轨迹图

可以预见，系统的动态性能与上一步使用不含偶极子的校正装置校正时的动态性能不会有明显的变化。因此，使用此校正装置的系统的动态性能是可以满足要求的。而根据偶极子的特性，此校正装置在低频段会提供大约 6(0.12 ÷ 0.02 = 6)倍于原系统增益的静态开环增益，使得系统的静态性能也可以满足最初的要求。因此，加入偶极子可以在不影响系统动态性能的情况下改善系统的静态性能。

图 4－42 的原点附近有一块根轨迹比较密集的区域，其放大图就是图 4－41。用"Zoom in X"和"Full View"按钮可以实现两图之间的相互转换。

同样，在图 4－42 中选中"Step"和"Bode"复选框，可以得到此时系统的响应曲线，校验最终的设计结果。

系统的阶跃响应曲线如图 4－43 所示。



图 4－43 校正后系统的阶跃响应曲线

可以看出，系统的阶跃响应曲线属于跟踪性能较好的振荡型曲线，快速性和稳定性都能满足要求。

系统的 Bode 图如图 4-44 所示。

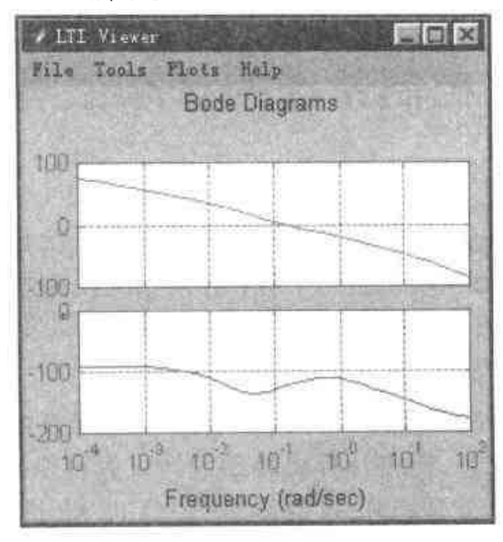


图 4-44 校正后系统的 Bode 图

从图中可以读出，此时系统的超调量 $\sigma\%$ 约为 25%，过渡过程时间 ts 不超过 1.2s。满足最初的要求。$\omega=0.02$ 处系统的增益接近 35dB，对应开环增益约为 60/s 与前边分析的数据也是相符的。因此，加入偶极子后系统的动态指标和静态指标都能满足要求，从图 4-42 第二行读出的 K 就是最后的校正装置。

小结：本例详细介绍了 Rltool 仿真环境的使用方法和技巧，第一次接触该环境的控制领域的专业人员尤其是刚开始学习控制理论的学生一定会深为其直观、快捷和方便而叹为观止。事实上，MATLAB 5.x 加强了有关图形界面设计的功能，经过一定的学习之后我们自己也可以编制出漂亮的图形仿真环境，不过遗憾的是这不是本书的主题，就不再介绍了。感兴趣的读者可以参看 MATLAB 自带的“GUI Layout”。

4.5 小 结

本章我们主要介绍了控制系统的校正指标和经验公式、控制系统开环频率特性设计、控制系统的 PID 校正和串联校正在 MATLAB 下的实现方法，最后我们介绍了一个 MATLAB 环境下的根轨迹设计工具——Rltool 仿真环境。Rltool 仿真环境充分体现了 MATLAB 5.x 版本的精髓，即操作完全可视化。因此，作为控制领域的研究者和工程技术人员来说，可以更方便的掌握其使用方法和技巧，把更多的精力放在有关控制系统的设计和仿真等问题上来。

MATLAB 在以传递函数为模型的控制系统校正中的应用就介绍到这里，下一章我们将讨论 MATLAB 在状态方程模型校正中的应用问题。

第五章　控制系统的状态空间设计方法

从20世纪30年代以来，基于状态空间模型的控制系统设计方法有了很大的发展。50～60年代，控制系统的研究者们提出了各种控制系统设计方法，其中影响较大、比较有代表性的包括极点配置设计方法、多变量系统的解耦控制以及以线性二次型性能(LQR)指标为基础的最优控制问题。

利用状态空间描述和校正线性定常系统的重要特点是校正方法的规范化。校正方法的规范化很适于计算机编程，使得计算机辅助设计成为可能。我们已经知道，在MATLAB环境下最基本的元素就是矩阵，这使得使用MATLAB进行状态空间模型的综合校正非常方便。关于状态空间的校正，应明确以下几点：

1. 校正的目标是什么。
2. 采用什么样的控制规律达到这个校正目标。
3. 线性定常系统(A,B,C,D)在什么条件下可以实现这个目标。
4. 实现校正目标的设计方法是什么。
5. 采用该设计方法所得的结果是否唯一。

应当指出的是，状态空间设计方法和频率域的设计方法并不是相互独立地发展的，它们之间有着千丝万缕的联系。对于同一个控制系统来说，可以采用传递函数描述，也可以采用状态空间描述。同样，对于同一个要求的性能指标，可以考虑采用频率域校正方法，也可以考虑状态空间的校正方法。

当然，这两种校正方法各有其优点。一般来说，大型系统的理论推导及证明问题往往需要从状态空间的角度入手，而控制器的设计及校正等问题又要在频率域下实现。而在实际的工程领域，航天技术、精密仪器、机械领域和制造业多采用状态空间设计方法，以化工为代表的流程工业多采用频率域设计方法。

在MATLAB环境下，使用状态空间设计方法和使用频率域设计方法都是非常方便的，本章我们主要讨论状态空间设计方法。

5.1　状态反馈与观测

对于控制系统的状态空间描述(A,B,C,D)，我们在本书第二章里已经详细讨论了它在MATLAB下的实现方式和可控性、可观性的判断。下一步，就是利用状态反馈矩阵对其进行综合校正。对于状态完全可控的系统，我们可以设计反馈校正装置R实现闭环极点的任意配置；而对于状态不完全可观的系统，就存在状态重构或称观测器设计的问题。本节将详细讨论这些问题并给出MATLAB实现。

5.1.1 极点配置

控制系统的各种特性及其各种品质指标很大程度上由其闭环系统的零点和极点的位置决定。极点配置问题就是通过对状态反馈矩阵的选择,使闭环系统的极点配置在所希望的位置上,从而达到一定的性能指标要求。

希望极点组的选择是一个确定校正目标问题。一般说,这是一个复杂的问题,是一个工程实践与理论相结合的问题。需要注意以下几条:

1. 对于 n 维系统,应当指定且只应当指定 n 个希望的极点。

2. 希望的极点可以是实数,也可以是按共轭对出现的复数;

3. 确定希望极点的位置,需要考虑极点和零点在复数平面上的分布,从工程实际出发加以解决。

对于 SISO 系统来说,其极点配置问题一般可参照下面的思路:

设控制系统的状态方程模型为(A,B,C,D),当引入状态反馈后,系统的控制信号为 $u = r - K^T X$,这里 r 是系统外部的输入;K 是一列向量,有时也称为反馈矩阵;此时闭环系统状态方程模型为:

$$\begin{cases} X = (A - BK^T)X + Br \\ Y = (C - DK^T)X + Dr \end{cases}$$

可以证明,当系统状态完全可控时,则可以通过状态反馈将系统的极点配置在复平面的任何位置上。

这里给出常用的两种配置系统闭环极点的方法以及相应的 MATLAB 实现,具体的证明请参阅相关书籍。

1. Gura-Bass 算法

假定期望的闭环极点为 $p_i, i = 1,2,\ldots,n$,则原系统的开环特征方程 $\alpha(s)$ 和闭环系统特征方程 $\beta(s)$ 分别可以写成:

$$\alpha(s) = \det(sI - A) = s^n + \sum_{i=0}^{n-1} \alpha_i s^i$$

$$\beta(s) = \prod_{i=1}^{n}(s - p_i) = s^n + \sum_{i=0}^{n-1} \beta_i s^i$$

若原系统状态完全可控,则状态反馈矩阵 K 可由下式求出

$$K^T = (\alpha - \beta)^T L^{-1} Q^{-1}$$

式中各参量的含义为:

$$(\alpha - \beta)^T = ((\alpha_0 - \beta_0), \quad \cdots, \quad (\alpha_0 - \beta_0))$$

$$L = \begin{bmatrix} \alpha_1 & \alpha_1 & \cdots & \alpha_1 & 1 \\ \alpha_1 & \alpha_1 & \cdots & 1 & \\ \vdots & \vdots & 1 & & \\ \alpha_1 & 1 & & & \\ 1 & & & & \end{bmatrix}$$

$$Q = (B, AB, \cdots, A^{n-1}B)$$

可见，若原系统状态完全可控，则 Q 的逆矩阵存在。而 L 是 Hankel 矩阵，因此总能解出状态反馈矩阵 K，因此能够将系统的闭环极点配置在复平面的任何一个位置。

对于 Gura-Bass 算法，MATLAB 控制系统工具箱中并没有给出相应的可以一步求解状态反馈矩阵的函数。不过从上面的分析中我们可以看出，状态反馈矩阵的表达式很简单，都是矩阵运算。而且细心的读者可能还记得，在第一章里我们曾经介绍过 MATLAB 环境下如何生成 Hankel 矩阵的方法。因此，我们可以自己编写用 Gura-Bass 算法求解状态反馈矩阵的 M 文件。

2. Ackermann 配置算法

设问题的描述与 Gura-Bass 算法相同。因为系统状态完全可控，所以 Ackermann 配置算法是先把系统的描述化为某种可控规范型，然后利用规范型的性质求解。具体的证明这里就不讨论了，只给出最后的结果：

$$K^T=(0,\cdots,0,1)Q^{-1}\beta(A)$$

各参量的含义与 Gura-Bass 算法相同，K 是所求的状态反馈矩阵。

MATLAB 提供了一条基于 Ackermann 配置算法求解系统状态反馈矩阵的函数 acker()，其基本调用格式为

$$K = ACKER(A,B,P)$$

其中 A，B 就是系统状态方程描述的参数矩阵；P 是期望的极点向量；K 就是所求的状态反馈矩阵。不过 MATLAB 同时警告说，该算法的数值稳定性并不是非常好，而且当系统的阶次高于 10 或系统的可控性矩阵接近奇异时，有可能引起算法的崩溃。不过 MATLAB 同时还指出，如果求得的闭环非零极点与期望的极点向量 P 的规定值相差超过 10% 的话，MATLAB 会提出警告。

一般说来，在 MATLAB 环境下，无论是应用 Gura-Bass 算法手工计算状态反馈矩阵 K 还是应用 acker() 函数计算，首先都应该对相同的状态可控性作出判断。否则就有可能得出错误的结果。尤其是在设计大型控制系统时，虽然 MATLAB 会提出警告，但最好还是掌握每一步计算的流程。

[例 5.1] 设某小球-滑板系统如图 5-1 所示。

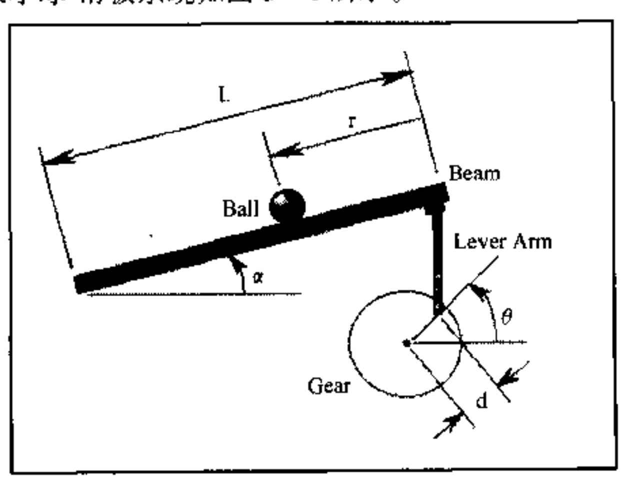


图 5-1 小球-滑板控制系统示意图

通过改变齿轮的角度 θ 可以达到控制小球的位置 r 的目的。其状态方程如下,其中小球质量 M ＝0.11 kg,小球直径 R＝0.015 m,杠杆偏移量 d＝0.03 m,重力加速度 g＝9.8 m/s^2,滑板长度 L＝1.0 m,小球的转动惯量 J＝9.99e－6 kgm^2,r 是小球的位置坐标,alpha 是滑板的角度,theta 是齿轮的角度。四个状态变量分别是小球位置 r、r 的一阶导数、滑板的倾角 alpha、alpha 的一阶导数。控制量 u 是齿轮的角度 θ,输出量是小球的位置 r。试根据其状态方程评价其运动特性,并分别根据 Cura-bass 算法和 Ackermann 算法配置其闭环极点,使其阶跃响应满足:过渡过程时间 t＜3s,超调量 σ%＜5%。

$$A=\begin{bmatrix} & 1 & & \\ & & \dfrac{-mg}{\left(\dfrac{J}{r^2}+m\right)} & \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} \qquad B=\begin{bmatrix}0\\0\\0\\1\end{bmatrix}$$

$$C=[1\quad 0\quad 0\quad 0] \qquad D=0$$

结果:未校正前系统的零极点和增益为:

```
z = Empty matrix: 0 - by - 1
p = 0   0   0   0
k = 7.0000
```

校正后系统的零极点和增益为

```
z = Empty matrix: 0 - by - 1
p =  - 80.0000     - 20.0000     - 2.0000 + 2.0000i     - 2.0000 - 2.0000i
k = 7.0000
```

状态反馈矩阵为

```
K = 1.0e + 003 *
     1.8286     1.0286     2.0080     0.1040
```

分析:一般来说,对于这种机械控制系统,不加反馈的开环控制很难获得满意的效果,必须要增加状态反馈矩阵。因此,本例的关键就在于将系统要求的性能指标转化为复平面上极点的位置。解决这一步后,相应的极点配置问题就可以通过 MATLAB 编程或 acker()函数来解决。在 MATLAB 环境下,还可以通过绘图来验证校正的结果。当然,在求解之前还要进行可控性的判断。

求解过程:本例的求解分为以下几步:

1. 原控制系统分析及期望性能指标与闭环极点间的转化

首先要分析一下原系统的零极点分布及阶跃响应的情况,看看它和期望的性能指标之间有什么差距。

在 MATLAB Editor/Debugger 下输入以下代码:

```
%系统参数初始化
m = 0.111;
R = 0.015;
```

```
g = -9.8;
J = 9.99e-6;
H = -m * g/(J/(R^2) + m);
%系统状态方程模型
A = [0  1  0  0
     0  0  H  0
     0  0  0  1
     0  0  0  0];
B = [0;0;0;1];
C = [1 0 0 0];
D = [0];
%原系统的零极点和增益
[z,p,k] = ss2zp(A,B,C,D,1)
%绘制其阶跃响应
T = 0:0.01:5;
U = 0.25 * ones(size(T));
[Y,X] = lsim(A,B,C,D,U,T);
plot(T,Y);
title('Ball - Beam System Step Response');
xlabel('Time - sec');
ylabel('Response - value')
```

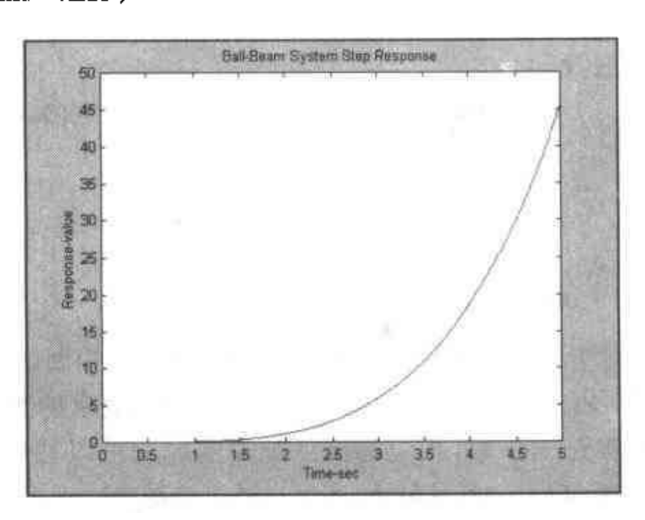


图 5-2 小球-滑板系统校正前阶跃响应

校正前系统的零极点和增益可参看上一步"结果"中的数据，可以看到，系统没有零点，极点均为零。这样的系统是非常不稳定的。从图 5-2 系统的阶跃响应曲线来看，在阶跃输入下其响应值趋于无穷大，根本谈不上与期望的性能指标作比较，必须采取状态反馈的校正手段。

然后要将期望的性能指标转化为复平面上极点的位置。这一步在第四章中有类似的例子，我们再复习一下。其思路就是根据经验公式和性能指标确定一对主导闭环极点，然后将非主导极点放在复平面上远离主导极点的地方。具体的经验公式请参看上一章的

4.4 节　在 MATLAB 下求得的主导极点为 $-2\pm j2$，因为原系统是四阶的，所以选取另外两个非主导极点为 -20 和 -80。

2. 判断可控性并分别按照 Gura－bass 算法和 Ackermann 算法配置其闭环极点

因为系统的四个极点都要配置到新的位置，所以必须对系统的可控性作出判断。只有状态完全可控的系统才能完成这一工作。否则就必须根据原来的物理系统对列出的状态方程作出调整。在上一步的窗口下输入以下代码：

```
%系统的阶次
n = length(A);
%状态可控性矩阵
Q = zeros(n);
Q(:,1) = B;
for i = 2:n
%求解状态可控性矩阵
    Q(:,i) = A * Q(:,i-1);
end
%可控性矩阵的阶次
m = rank(Q);
%判断系统是否可控
if m = n
%系统完全可控
    disp('System State Variables can be totally controlled');
else
%系统不完全可控
    disp('System State Variables cannot be totally controlled');
    disp('The rank of System Controllable Matrix is:');
    m
end
```

结果为：

```
System State Variables can be totally controlled
```

因此，系统状态完全可控，可以对系统的四个极点进行任意位置的配置。下面首先按照 Gura-bass 算法编程：

```
%系统的期望闭环极点
p1 = -2 + 2i;
p2 = -2 - 2i;
p3 = -20;
p4 = -80;
P = [p1,p2,p3,p4];
%Gura-Bass 校正算法
alphaS = poly(A);
betaS = conv([1, -p1],conv([1, -p2],conv([1, -p3],[1, -p4])));
i = length(alphaS): -1:2;
diff = alphaS(i) - betaS(i);
```

```
L = hankel([alphaS(length(alphaS) - 1: - 12)';1]);
K = diff * inv(L) * inv(Q);
disp('The state - feedback Matrix is:');
K
disp('The close - loop poles are:');
eig(A - B * K)
```

结果为:

```
The state - feedback Matrix is:
K  =  1.0e+003  *
      1.8286       1.0286       2.0080       0.1040
The close - loop poles are:
ans  = - 80.0000
       - 20.0000
       - 2.0000 + 2.0000i
       - 2.0000 - 2.0000i
```

然后调用 acker()函数,用 Ackermann 算法配置其闭环极点:

```
K = acker(A,B,P);
```

结果为:

```
K  = 1.0e+003  *
     1.8286       1.0286       2.0080       0.1040
```

从以上的结果中可以看出,用 Gura-bass 算法编程和直接使用 acker()函数得到的结果是完全一样的。不过一般控制理论的入门书籍中讲述的都是 Gura-bass 算法,我们这里给出该算法的源程序也是为了使读者更加清晰的掌握状态反馈校正的设计思路和实现方法。如果是控制领域的工程技术人员的实际应用,如果阶次低于 10 阶的话,还是直接调用 acker()函数更加简便。

3. 闭环阶跃响应评价

为了检验校正的结果,我们可以绘制其闭环阶跃响应曲线如图 5-3 所示。

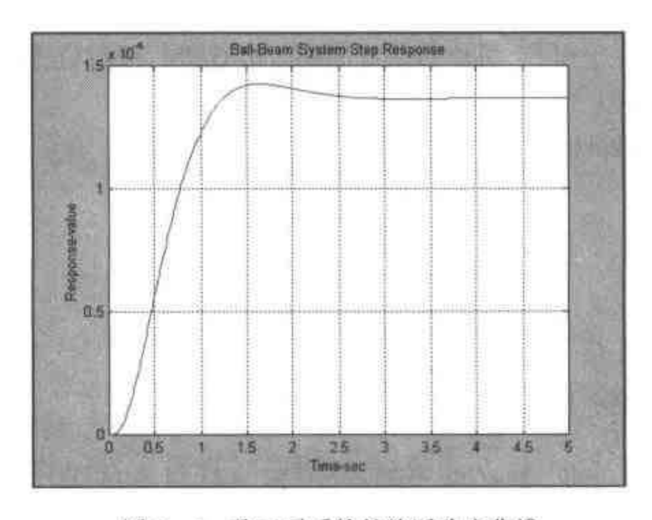


图 5-3　校正后系统的阶跃响应曲线

相应的代码为：

```
T = 0:0.01:5;
U = 0.25 * ones(size(T));
[Y,X] = lsim(A - B * K,B,C,D,U,T);
plot(T,Y);
title('Ball - Beam System Step Response');
xlabel('Time - sec');
ylabel('Response - value');
grid;
```

从图 5-3 中可以看出，系统的快速性很好，过渡过程时间不超过 2.5s。并且响应过程中只振荡了一次，超调量也非常小，基本满足最初的设计要求：过渡过程时间 $t<3s$，超调量 $\sigma\%<5\%$。

不过如果仔细阅读了源代码的读者就会发现，系统的输入量是 0.25 倍的阶跃函数，但阶跃响应的稳态值仅为 1.4e-4，相差了三个数量级。出现这种情况的原因是因为全状态反馈自身的特性。与其他反馈手段不同，全状态反馈不但要将输出量反馈回去，还要将所有的状态变量反馈到输入端。而输入端的输入量只有一个，静态无差是指输出量反馈回来后与该输入量差值为零。这么多的状态变量都反馈回来，还要做到差值为零，输出量 Y 势必要发生变化，其减小或增大与状态反馈矩阵 K 有关。

本例中 K 是 10 的三次方数量级的，因此输出量 Y 也就减小相应的倍数。为了达到静态无差，必须在输出端设计一种装置，改善这种状况。

4. 求解参考输入并检验结果

改善这种状况的方法就是将系统的输入函数乘以一个参考量，得到系统的参考输入(Reference Input)，以此参考输入与反馈回来的值进行比较得到控制量。其系统结构图如图 5-4 所示。

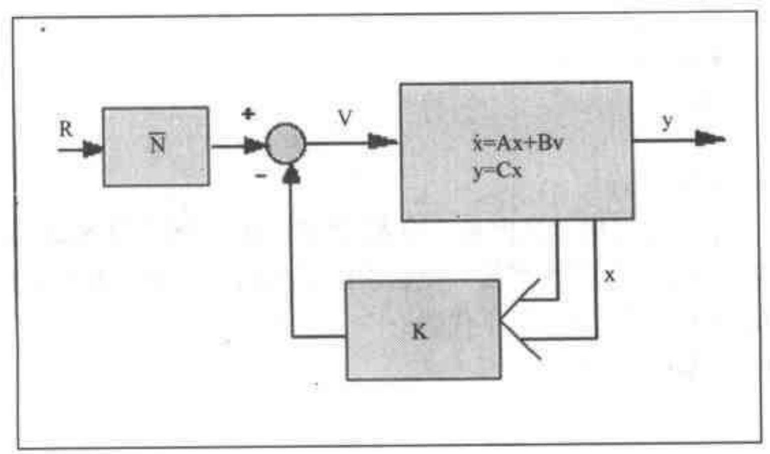


图 5-4 包含参考输入的系统结构图

关于参考输入倍数的计算请参看有关的控制理论书籍。MATLAB 下没有给出相关的函数，但密歇根大学(University of Michigan)的 Yanjie Sun 博士给出了一个 M 函数 rscale()，可以求解系统参考输入。

这个函数属于自由软件，读者可以在相关网站上下载到。下面给出了该函数的源代

码，读者可以将下面的代码输入到计算机，保存在 MATLAB 的“BIN”目录下就可以使用了。事实上，MATLAB 的相关网站上有许多可供自由下载的 M 函数，我们在附录中会作出相应的介绍。rscale()函数代码为：

```
function[Nbar] = rscale(A,B,C,D,K)
% Given the single - input linear system:
%        .
%        x  =  Ax  +  Bu
%     y  =  Cx  +  Du
% and the feedback matrix K,
%
% the function rscale(A,B,C,D,K) finds the scale factor N which will
% eliminate the steady - state error to a step reference
% using the schematic below:
%
%                    / ------ \
%    R          +          u  | .            |
%         ---> N ---> () ----> | X = Ax + Bu |--> y = Cx ---> y
%              -|          \ ------/
%               |                    |
%               | <--- K <---|
%
%8/21/96 Yanjie Sun of the University of Michigan
%          under the supervision of Prof. D. Tilbury
%
s = size(A,1);
Z = [zeros([1,s]) 1];
N = inv([A,B;C,D]) * Z';
Nx = N(1:s);
Nu = N(1 + s);
Nbar = Nu + K * Nx;
```

有了这部分代码，就可以计算出参考输入的倍数 Nbar，然后用 Nbar 乘以 B 矩阵就可以得到系统的参考输入。使用参考输入重新求解系统的闭环阶跃响应，就可以让输出量严格跟随输入量的变化了。输入以下代码：

```
Nbar = rscale(A,B,C,D,K);
disp('The Reference Input Value is:');
Nbar
T = 0:0.01:5;
U = 0.25 * ones(size(T));
[Y,X] = lsim(A - B * K,B * Nbar,C,D,U,T);
plot(T,Y)
title('Ball - Beam System Step Response with Reference Input');
xlabel('Time - sec');
```

```
ylabel('Response - value');
grid;
```

则系统的参考输入倍数为：

```
The Reference Input Value is
Nbar = 1.8286e+003
```

当然，用输入量的稳态值除以输出量的稳态值也能得到这个结果。不过那就是一种意义并不很明确的补偿办法，而不是理论分析得出的结果了。

相应的系统闭环阶跃响应如图 5-5 所示。

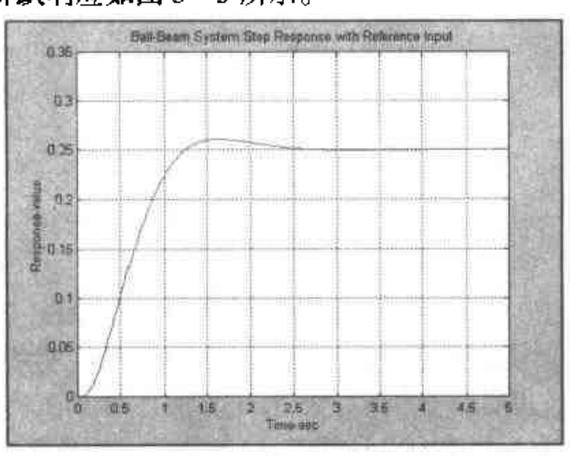


图 5-5 加入参考输入后系统的阶跃响应

从图 5-5 中可以看到，系统的稳态值与阶跃函数的输入值相同，均为 0.25。并且过渡过程时间和超调量与图 5-3 相比均未发生变化，仍旧满足过渡过程时间 $t<3s$，超调量 $\sigma\%<5\%$ 的要求。这一点很好理解，因为仅仅是输入端乘以了一个常数，系统的结构和闭环极点均未发生变化，主导极点还在原来的位置。因此，系统的动态性能指标也不应该有变化，只是稳态值提高了三个数量级。

小结：本例详细讨论了在状态空间描述下，使用 Gura-bass 算法和 Ackermann 算法配置系统闭环极点的思路和具体的 MATLAB 实现方法。并且根据实际情况，给出了一个实现参考输入的 M 函数。

从校正的效果来看，系统的各项性能指标还是能够令人满意的。对比上一章的根轨迹校正方法，同样是配置闭环极点，采用根轨迹校正时必须改变系统的结构(增加校正装置)。而配置状态方程的闭环极点则只需设计相应的状态反馈矩阵，状态变量还是原来的四个量，也就是说，系统的结构没有发生改变。对于实际的工程领域来说，有时候这一点是非常重要的。因为有些封装牢固或比较精密的仪器是不允许改变结构的，此时就显示出状态空间极点配置方法的优越性了。

并且，在 MATLAB 环境下，配置状态方程的闭环极点只需一条 acker()关键语句即可，不必像根轨迹校正或 PID 校正那样绘制大量图形。数学基础和理论完善，计算机仿真软件实现比较简单，这也是以状态方程为基础的现代控制理论蓬勃发展的重要原因。当然，频率域设计也有其优点，特别是对系统的机理了解不是很清楚、控制指标要求不高时，采用频率域的方法还是非常有效的。总而言之，这两种方法互为补充，互相渗透，都是控制系统设计的重要手段。

MATLAB 中还有一条配置状态方程闭环极点的语句 place(),并且它是为 MIMO 系统设计的(acker()函数是为 SISO 系统设计的)。其基本调用格式与 acker()函数类似,但该函数有两个可选的返回值 PREC 和 MESSAGE。PREC 表示求得的 A - B * K 的特征值与期望的闭环极点之间的相似程度;如果求得的极点与期望的闭环极点之间相差超过10%的话,相关的警告信息就存在 MESSAGE 里面。

虽然 place()函数是为 MIMO 系统设计的,但同样适用于 SISO 系统(为 SISO 系统设计的函数一般不适用于 MIMO 系统)。例如:对于例 5.1,在其工作空间(Workspace)的 MATLAB Command Window 下输入:

```
[K,PREC,MESSAGE] = PLACE(A,B,P)
```

结果为

```
K =    1.0e+003 *
       1.8286      1.0286      2.0080      0.1040
PREC =      15
MESSAGE = ''
```

求得的状态反馈矩阵 K 与 acker()函数的结果一样。PREC 表示的是求得的极点与期望的闭环极点相同的小数位数,15 是一个很高的数字了,说明两者之间的差别非常小。MESSAGE 里没有信息,说明对于本例的数值,该算法是稳定的。一般说来,PREC 很高,MESSAGE 里就没有信息。而 PREC 较低时,一般是系统的参数矩阵接近奇异或状态不完全可控,此时系统会在 MESSAGE 里给出信息:

```
MESSAGE = 'Matrix A is ill conditioning'
```

表明矩阵 A 为病态矩阵。此时实现全极点状态反馈是不可能的,一般要改变系统的结构,从而改变矩阵 A 的参数。如果 MESSAGE 里给出信息是系统接近不完全可控,则需要重新选择状态变量,使新的状态变量完全可控。

5.1.2 状态观测器

上一小节中我们叙述了状态完全可控的系统(A,B,C,D)可以通过状态反馈任意配置闭环极点。为了实现状态反馈,需要系统所有的状态信息。但是系统的所有状态不一定都能测量到,这就造成了状态反馈在物理实现上的困难。也就是说,即使理论上证明了系统状态完全可控,能实现全极点状态反馈,也必须根据系统的实际情况来作出选择。这就提出了状态重构问题。

状态重构问题的核心,就是重新构造一个系统,利用原系统可以直接测量的变量如输出量 y 和输入量 u 作为它的输入信号,并使其输出信号 $x'(t)$在一定指标下和原系统的状态变量 $x(t)$等价。通常把 $x'(t)$叫做 $x(t)$的状态重构或状态估计,而把实现状态重构的系统叫做观测器。此时系统结构图如图 5 - 6 所示。

$x'(t)$和 $x(t)$间的等价性指标常常采用渐进等价的指标,即:

$$\lim_{t\to\infty}\tilde{x}(t) = \lim_{t\to\infty}[x(t) - x'(t)] = 0$$

其中 $\tilde{x}(t) = x(t) - x'(t)$称为观测误差。

该观测结构的基本方程为:

$$\dot{x}' = (A - LC)x' + Bu + Ly$$

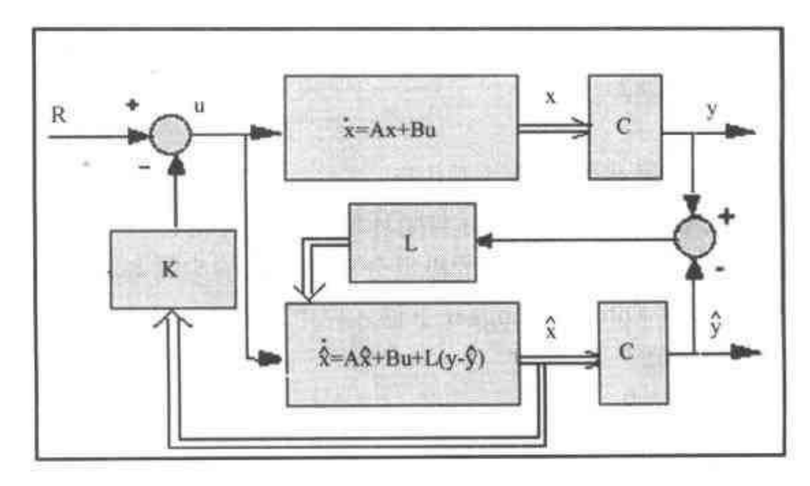


图 5-6 加入观测器后的系统结构图

可见,该基本观测器可以任意配置极点的充分必要条件是矩阵对(A,C)完全可观测。完全可观测系统的基本观测器的极点配置设计可以仿照完全可控系统用状态反馈进行极点配置的方法进行。请看下例。

[例 5.2]　某电磁振荡系统如图 5-7 所示。

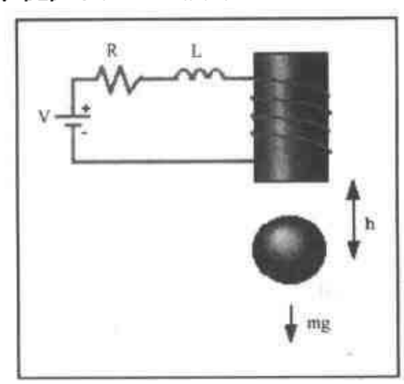


图 5-7 电磁振荡系统示意图

状态方程已知,三个状态变量分别是小球与线圈的位置变化量 Δh、Δh 的一阶导数和电流的变化量 Δi。输入量是电源电压 V,输出量是 Δh。其中小球质量 $M = 0.05$ Kg,电磁常数 $K = 0.0001$,电感 $L = 0.01$ H,电阻 $R = 1\Omega$,重力加速度 $g = 9.81\ m/s^2$。

该电磁振荡系统的控制要求为:

1. 设计状态反馈矩阵 K 及参考输入,满足过渡过程时间 $t < 0.5s$,超调量 $\sigma\% < 15\%$。

2. 假设系统的某些状态变量无法直接测量,试选择合适的参数设计状态观测器,给出相应的观测误差。

结果:求得的状态反馈矩阵为:

K =　-280.7143　　-7.7857　　-0.3000

观测器矩阵为:

$$A = \begin{bmatrix} 0 & 1 & 0 \\ 980 & 0 & -2.8 \\ 0 & 0 & -100 \end{bmatrix} \qquad B = \begin{bmatrix} 0 \\ 0 \\ 100 \end{bmatrix}$$

$$C = [1 \quad 0 \quad 0] \qquad D = 0$$

```
L = 1.0e+004 * 0.0203
               1.1282
               0
```

求解过程:本例的解题步骤分为以下几步:

1. 原控制系统分析及期望性能指标与闭环极点间的转化

与例 5.1 相同,首先要分析原系统的极点分布和阶跃响应的基本情况,判断采用何种校正手段。在 MATLAB Editor/Debugger 下输入以下代码:

```
%系统状态方程描述
A = [0   1   0
     980 0   -2.8
     0   0   -100];
B = [0
     0
     100];
C = [1  0  0];
D = 0;
%系统的极点分布情况
poles = eig(A);
%闭环阶跃响应曲线绘制
t = 0:0.01:2;
u = 0.25 * ones(size(t));
x0 = [0.005  0  0];
[y,x] = lsim(A,B,C,D,u,t,x0);
h = x(:,2);
plot(t,h);
title('Magnetic System Step Response');
xlabel('Time - sec');
ylabel('Response');
grid;
```

求得系统的闭环极点为

```
ans =  31.3050
      -31.3050
      -100.0000
```

可以看到,系统有一个右半平面的极点 31.3050,因此未加校正前系统的阶跃响应是不稳定的。从图 5-8 的曲线来看也是如此,曲线以极快的速度最终趋于无穷。考虑到状态方程已知,可用状态反馈的方法将闭环极点配置在任意的位置。

根据性能指标的要求,过渡过程时间 $t<0.5s$,超调量 $\sigma\% < 15\%$,求得期望的闭环主导极点为 $-10 \pm j10$(具体求解方法可参看例 5.1 或第四章 4.4 节)。因为系统是三阶的,选取另一个非主导极点为 -50。

2. 求解相应的状态反馈矩阵及参考输入

通过例 5.1 我们知道,想要让输出量跟随输入量的变化必须增加参考输入。因此,首

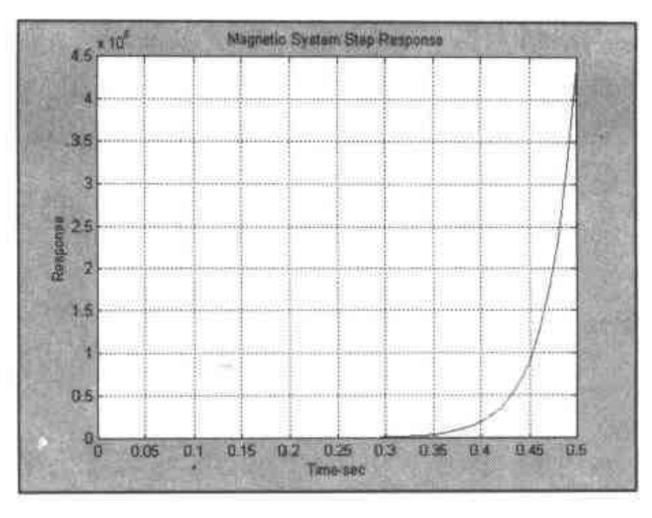


图 5-8 原系统闭环阶跃响应曲线

先判断可控性，然后根据期望闭环极点求解状态反馈矩阵，最后求出参考输入。紧接上一步，输入代码：

```
%期望闭环极点
p1 = -10 + 10i;
p2 = -10 - 10i;
p3 = -50;
P = [p1,p2,p3];
%判断可控性
n = length(A);
Q = zeros(n);
Q(:,1) = B;
%求解可控性矩阵
for i = 2:n
    Q(:,i) = A * Q(:,i-1);
end
m = rank(Q);
if m = n
    K = place(A,B,P);
else
    disp('System State Variables cannot be totally controlled');
    disp('The rank of System Controllable Matrix is:');
    m
end
%求解参考输入
Nbar = rscale(A,B,C,D,K);
lsim(A - B * K,B * Nbar,C,D,u,t);
grid;
```

求得的校正后系统的阶跃响应曲线如图 5-9 所示。

相应的状态反馈矩阵为：

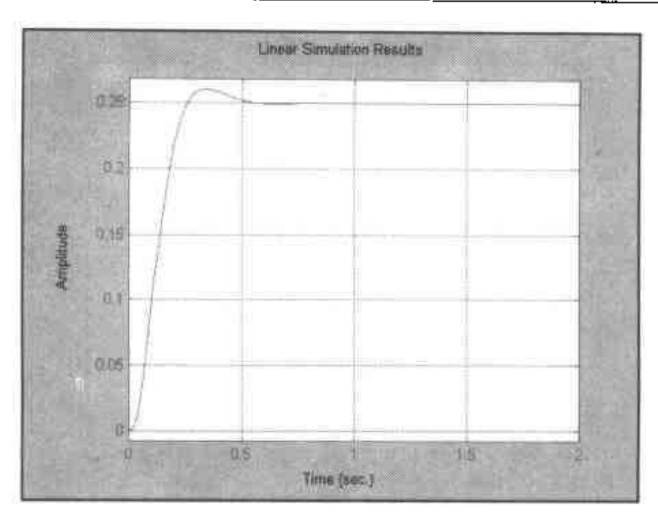


图 5－9 校正后系统的阶跃响应曲线

K = －280.7143 －7.7857 －0.3000

参考输入倍数为：

Nbar = －35.7143

调用 place()函数返回的精度为：

place: ndigits = 15

表明配置的极点与期望闭环极点之间的差距非常小。从图 5－9 的阶跃响应曲线来看，系统的快速性和稳定性都满足最初要求的过渡过程时间 $t<0.5s$，超调量 $\sigma\%<15\%$（注意，这里为了观察的更清楚，输入量选用的是 0.25 倍的阶跃函数）。并且，系统在达到稳态值之前只振荡了一次。

3．求解状态观测器

对于实际的控制系统来说，即使状态完全可控，也很难做到系统的每个状态变量都能够直接用仪器测量。因此，有必要设计系统的状态观测器，通过重构的等价状态变量来构成状态反馈。一般说来，如果观测器构造的比较好，使用重构的等价状态变量进行状态反馈和使用原状态变量进行状态反馈没有什么太大的区别。因此，我们用观测误差曲线来评价观测器的性能。

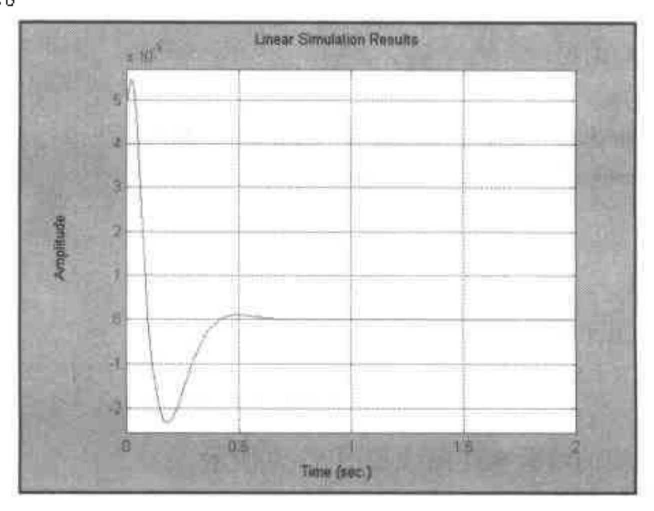


图 5－10 观测器误差响应曲线

图 5－10 为系统的观测器误差曲线，其纵坐标的数量级为 10^{-3}，表明误差在一个很小的范围内振荡，观测器能够反映状态变量的变化。

相应的状态变量的观测值响应曲线见图 5－11，由于观测误差的数量级为 10^{-3}，因此可以认为重构的状态变量能够反映原状态变量的变化。

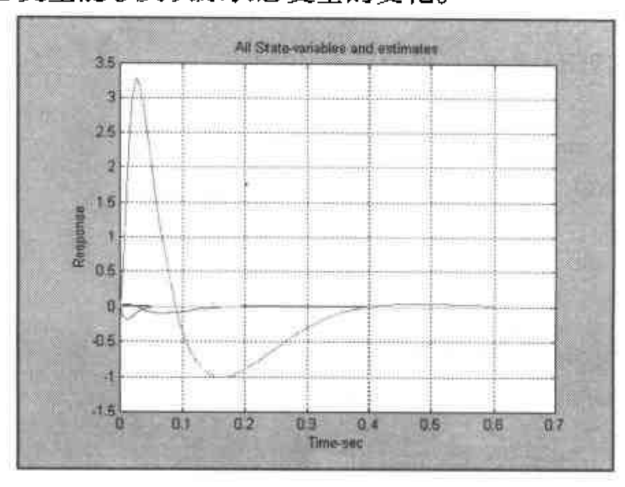


图 5－11 各状态变量的观测值响应曲线

在求解观测器时，首先要判断系统的可观性，然后配置观测器的极点。如果系统完全可观的话，我们希望观测器的响应要快于原系统的响应。因此，将观测器的三个极点均配置在－100 左右。具体的数字可以自己选择，其原则是在实际系统允许的情况下尽可能的靠近复平面的左边。紧接上一步，输入以下代码：

```
%系统观测器极点
op1 = -100;
op2 = -101;
op3 = -102;
%求解系统可观性矩阵
n = length(A);
Q = zeros(n);
Q(1,:) = C;
for i = 2:n
    Q(i,:) = Q(i-1,:) * A;
end
m = rank(Q);
if m = n
%求解观测器增益矩阵
L = place(A',C',[op1 op2 op3])';
%重构的系统及状态变量
At = [A - B*K          B*K
     zeros(size(A))       A - L*C];
Bt = [    B*Nbar
     zeros(size(B))];
Ct = [    C           zeros(size(C))];
```

```
%输出量观测误差
lsim(At,Bt,Ct,D,zeros(size(t)),t,[x0 x0])
grid;
%各状态变量观测误差
t=0:0.005:0.6;
[e,x] = lsim(At,Bt,Ct,D,zeros(size(t)),t,[x0 x0])
plot(t,x);
title('All State - variables and estimates');
xlabel('Time - sec');
ylabel('Response');
grid;
else
    disp('System State Variables cannot be totally Observed');
    disp('The rank of System Observable Matrix is:');
    m
end
```

求得的观测器增益矩阵为

```
L = 1.0e+004 *    0.0203
                  1.1282
                  0
```

相应的输出量观测误差和各状态变量观测值参看图5-10和图5-11。从这两幅图中可以看出，虽然误差响应有一定的振荡，但在0.5s之前都已经基本回到了零点。也就是说，0.5s以后重构的等价状态变量，已经可以代替原来的状态变量实现全状态反馈了。此时再根据第2步设计的状态反馈矩阵K进行运算，就可以把系统的闭环极点配置在复平面上希望的位置了。

小结：本例一方面复习了状态反馈矩阵的设计方法，另一方面又着重介绍了状态观测器的设计和MATLAB实现。从控制的效果来看，还是比较满意的。在控制理论的实际应用中，观测器的设计是非常重要的环节。一方面如前所述，系统的状态变量无法完全直接测量；另一方面有的系统的状态方程就是由传递函数转化得到的，其状态变量没有对应的物理意义，只能构造状态观测器来实现状态反馈。当然，这样做的前提是，系统的状态变量既是完全可控的又是完全可观的。

5.2 解耦控制

在多变量系统中，不同的输入和输出之间存在着耦合，即系统的第一个输入量不但会对第一个输出量产生影响，而且还会影响到其他的输出量。这样就造成了控制系统设计和实际操作的困难。因此，控制领域的工程人员就提出了解耦的思想，试图把多变量系统分解为多个单变量系统。

解耦控制又称为互不影响控制、一对一控制。最早的工作是由E.G.Eilbert完成的，当时称为Morgan问题。解耦问题是多输入量多输出量(MIMO)线性定常系统综合理论

的一个重要组成部分。其目的是寻找合适的控制规律使闭环控制系统实现一个输出分量仅仅受一个输入分量控制，而且不同的输出分量受不同的输入分量控制

假设多变量系统的模型描述为(A,B,C,D)，其中 A 矩阵是 $n\times n$ 维，B 矩阵是 $n\times p$ 维，C 矩阵是 $m\times n$ 维，D 矩阵是 $m\times p$ 维。在解耦系统中还要求 $m=p$，这是为了保证一个输入量对应一个输出量。则相应的解耦闭环传递函数为：

$$G(s)=(C-DF)(sI-A+BF)^{-1}BR+DR$$

其中 R 是 $m\times m$ 维输入变换矩阵，F 是 $m\times n$ 维状态反馈矩阵。

如果求得的变换传递函数 G(s)是对角的非奇异矩阵，则该系统是动态解耦的系统；若 G(0)是对角的非奇异矩阵，且系统是稳定的，则称该系统为静态解耦的系统。R 和 F 矩阵的求解可以参看相关的控制理论书籍，这里不加证明的给出其表达式：

$$R=D_0^{-1}$$

$$F=D_0^{-1}L$$

$$D_0=\begin{bmatrix} d_1^T \\ \vdots \\ d_m^T \end{bmatrix}=\begin{bmatrix} c_1^T & A^{\alpha_1-1} & B \\ & \vdots & \\ c_m^T & A^{\alpha_m-1} & B \end{bmatrix} \qquad L=\begin{bmatrix} c_1^T & A^{\alpha_1} \\ & \vdots \\ c_m^T & A^{\alpha_m} \end{bmatrix}$$

其中 a_i 称作系统的解耦阶常数，其定义为：

$$a_i\begin{cases} \min[k \mid c_i^T A^{k-1}B\neq 0, & (1\leqslant k\leqslant n)] \\ n(\text{当 } c_i^T A^r B=0, r=0,1,2\ldots,n-1) \end{cases}$$

c_i^T 是 C 矩阵的第 i 个行向量

解耦后的系统还可以进行 a 阶极点配置，具体的公式推导和证明这里就不讨论了。关于解耦控制的 MATLAB 实现请看下例：

[例 5.3]　某多输入多输出系统状态方程如下　其控制要求为：

1. 试求解其传递函数矩阵，并讨论系统的运动情况

2. 设计解耦控制器，有可能的话进行极点配置。

结果：原系统的传递函数矩阵为：

$$A=\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ -1 & -1 & -3 \end{bmatrix} \qquad B=\begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix}$$

$$C=\begin{bmatrix} 1 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad D=\begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

```
num1 =    0     1.0000     3.0000    -0.0000
          0     0         -1.0000    -0.0000
den1 =    1     3          1          0
num2 =    0     0          1.0000     0
          0     1.0000    -0.0000     0
den2 =    1     3          1          0
```

即
$$G(s)=\frac{1}{s^3+3s^2+s}\begin{bmatrix} s^2+3s & s \\ -s & s^2 \end{bmatrix}$$

解耦后系统的传递函数矩阵为：

```
numd1 =   0     1        0         0
          0     0        0         0
dend1 =   1     0        0         0
numd2 =   0   0.0000  -0.0000   -0.0000
          0   1.0000     0         0
dend2 =   1     0        0         0
```

即

$$G(s)=\frac{1}{s^3}\begin{bmatrix} s^2 & 0 \\ 0 & s^2 \end{bmatrix}=\frac{1}{s}\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

相应的输入变换矩阵 R 和状态反馈矩阵 F1 为:

```
R =   1    0
      0    1
F1 =   0    0    1
      -1   -1   -3
```

分析:对于多输入多输出系统,一般都要进行解耦控制或者对角化处理。对于本例,系统的状态方程模型已知,采用上边分析的反馈解耦的方法比较好。并且所有相关的运算都是矩阵运算,特别适用于 MATLAB 求解。不过这种反馈解耦的方法求得的解耦系统一般都是不同阶次的积分器类型的,真正要投入使用,必须要对相关的闭环极点位置进行配置,以获得期望的性能指标。

求解过程:

本例的解题步骤分为以下几步:

1. 求解原系统的传递函数及输入对输出的阶跃响应

首先要根据系统的传递函数判断是否解耦,然后根据其不同输入与输出之间的阶跃响应曲线进行验证。在 MATLAB Editor/Debugger 下输入以下代码:

```
A =    [0     0     0;
        0     0     1;
       -1    -1    -3];
B =   [1     0;
       0     0;
       0     1];
C =   [1     1     0;
       0     0     1];
D = zeros(2,2);
%求解传递函数矩阵
[num2,den2] = ss2tf(A,B,C,D,2);
[num1,den1] = ss2tf(A,B,C,D,1);
%绘制不同输入量的阶跃响应曲线
t = 0:0.1:10;
u = ones(size(t));
step(A,B,C,D,1,t);
step(A,B,C,D,2,t);
```

相应求得的系统传递函数矩阵请参看上一部分“结果”中的数据。

从“结果”的传递函数矩阵可以看出，系统的耦合比较严重。两个输入量对两个输出量的传递函数均不为零，因此相应的响应曲线也不为零，相互之间存在影响。图 5－12 是系统对第一个输入量的阶跃响应曲线，其上半部分是第一个输出量的响应曲线，下半部分是第二个输出量的响应曲线。图 5－13 是系统对第二个输入量的阶跃响应曲线，其结构与图 5－12 相同。

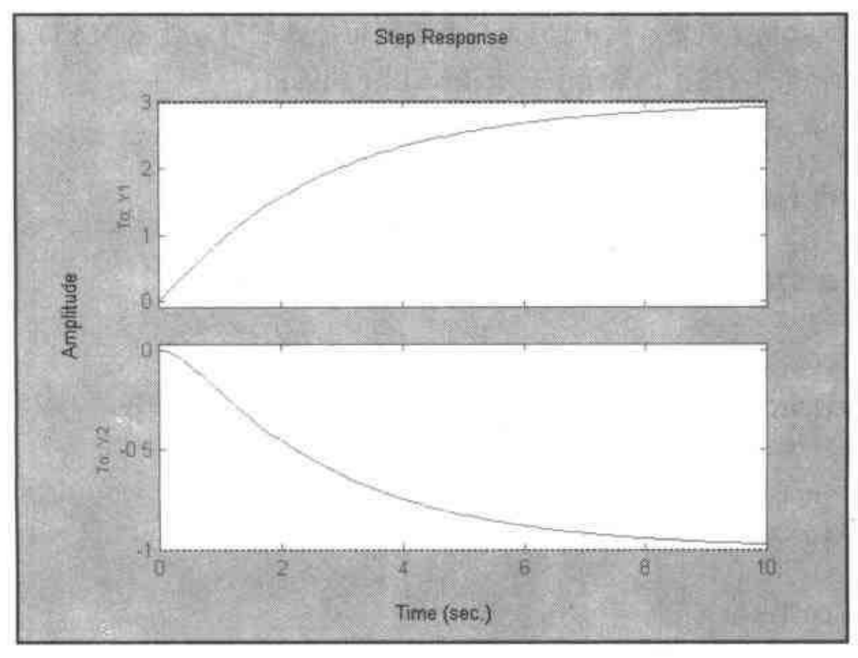


图 5－12　系统对第一输入量的阶跃响应曲线

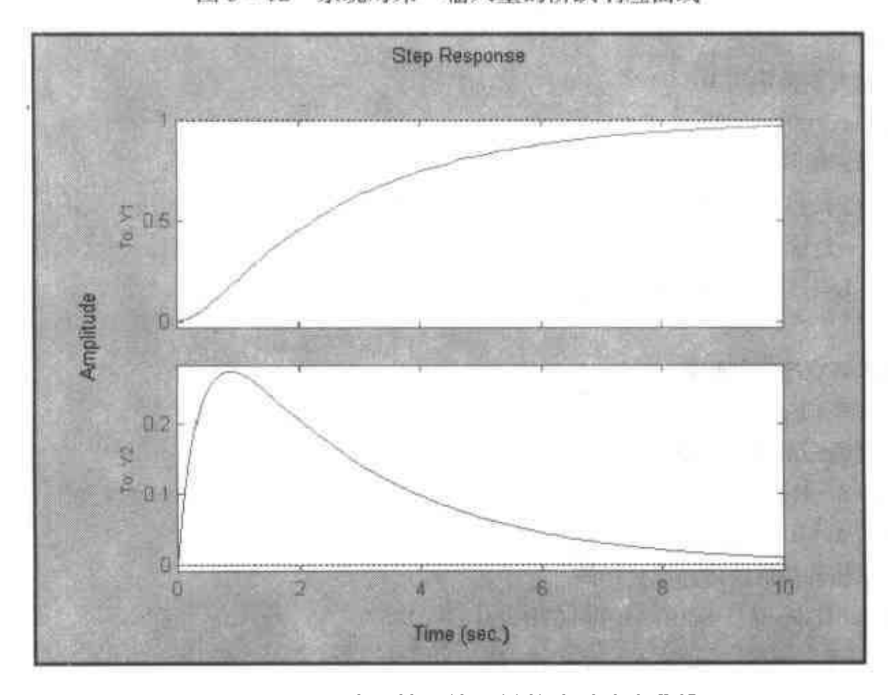


图 5－13　系统对第二输入量的阶跃响应曲线

从这两张图中也可以看出，输出量与输入量之间存在耦合，任一个输入量对两个输出量均有影响。如果不设计解耦控制器，这样的系统是很难控制的，其计算量随阶次的升高

以指数形式增加。

2. 解耦控制器设计

解耦控制器设计的关键是求解解耦阶常数 α_i。有了 α_i 之后，输入变换矩阵 R 和状态反馈矩阵 F 就非常容易求解了。当然，严格说来，在求解解耦控制器之前还应该判断系统的可控性和可观性，如果系统不完全可控可观的话是无法进行解耦控制器的设计的。为了节省空间起见这里就略去了，因为判断系统可控性和可观性的 MATLAB 程序在前文已经有较详细的叙述了。紧接上一步，输入以下代码：

```
[m,n] = size(C);
D0 = C(1,:) * B;
%解耦阶常数 a
a(1) = 0;
%求解解耦阶常数 a、矩阵 D0
for i = 2:m
for j = 0:n - 1
D0 = [D0;C(i,:) * A^j * B];
if rank(D0) = = i,
a(i) = j;
break;
else,
D0 = D0(1:i - 1,:);
    end
end
end
%输入变换矩阵 R
R = inv(D0);
%求解矩阵 L
L = C(1,:) * A^(a(1) + 1);
for i = 2:m
    L = [L;C(i,:) * A^(a(i) + 1)];
end
%求解状态反馈矩阵
F1 = R * L;
%解耦后系统的状态方程
B1 = B * R;
A1 = A - B * F1;
%系统解耦后的传递函数矩阵
[numd1,dend1] = ss2tf(A1,B1,C,D,1);
[numd2,dend2] = ss2tf(A1,B1,C,D,2);
%解耦后系统对两个输入量的阶跃响应
step(A1,B1,C,D,1,t);
step(A1,B1,C,D,2,t);
```

解耦后系统对第一输入量的阶跃响应曲线如图 5 - 14 所示。

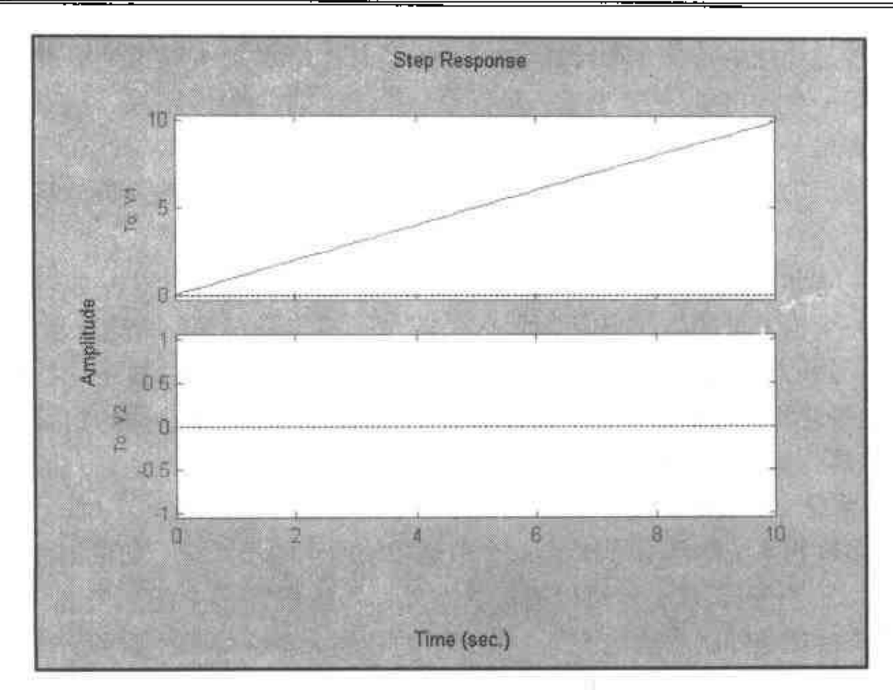


图 5-14 解耦后系统对第一输入量的阶跃响应曲线

此时第一输入量仅对第一输出量有影响，对第二输出量的影响为零，实现了对第一输入量的解耦。

解耦后系统对第二输入量的阶跃响应曲线为如图 5-15 所示。

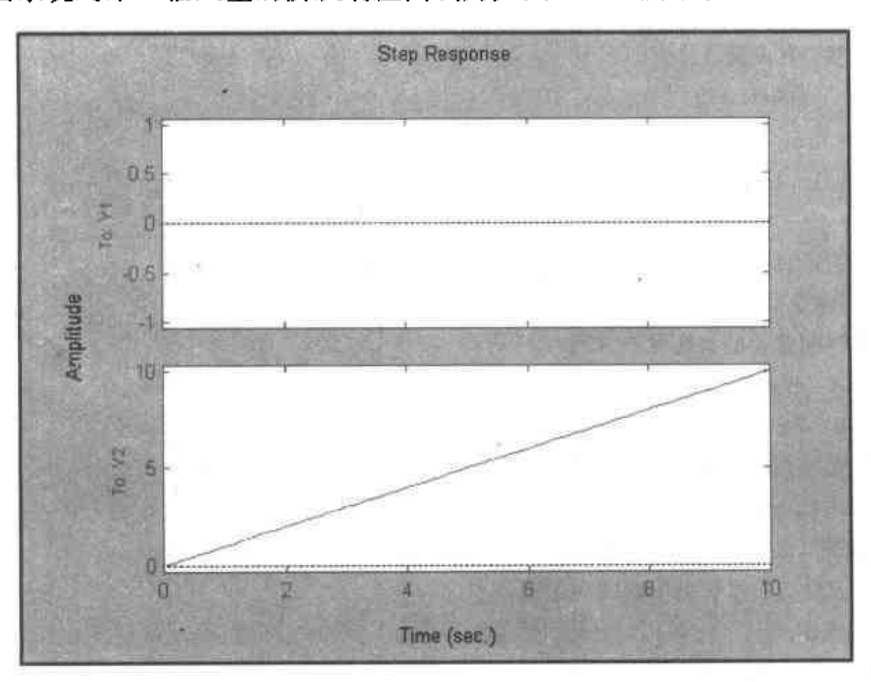


图 5-15 解耦后系统对第二输入量的阶跃响应曲线

同样，此时第二输入量仅对第二输出量有影响，对第一输出量的影响为零，实现了对第二输入量的解耦。

相应求得的解耦后系统传递函数矩阵和状态反馈矩阵 F1 请参看上一部分“结果”中

的数据。该传递函数矩阵是非奇异的对角矩阵，因此系统已经实现完全解耦。第一个输入量仅控制第一个输出量，第二个输入量仅控制第二个输出量。不过还应该看到，该矩阵的对角元素都是一阶积分器，其作用是对输入量作无穷积分。因此，如果没有限制的话，该系统是不稳定的。并且，由于两个传递函数相同，系统对第一个输入量的响应和第二个输入量的响应完全相同。

从图 5－14 和图 5－15 的阶跃响应曲线上我们也验证了这种说法。一方面，第二个输出量对第一个输入量的阶跃响应曲线为零，第一个输出量对第二个输入量的阶跃响应曲线也为零；另一方面，第一和第二个输出量对各自输入量的响应曲线是一条斜率为 1 的直线，这正是一阶积分器的阶跃响应曲线。除了特殊情况下，一般这样的相同是不能投入使用的，应该进行解耦系统的极点配置

3．解耦系统 a 阶极点配置

多输入多输出系统解耦后形成了 m 个单输入单输出的子系统。如果第 i 个子系统的传递函数是 α_i 阶积分器，那么就可以通过状态反馈矩阵配置 α_i 个闭环极点，称为 a 阶极点配置。状态反馈矩阵的求法与 5.1.1 节介绍的相同。本例都是一阶积分器，所以这两个子系统各可以配置一个极点。我们将第一个子系统的极点配置在 －2，第二个子系统极点配置在 －3。紧接上一步，输入以下代码：

```
%期望的闭环极点
beta1 = -2;
beta2 = -3;
beta = [beta1,beta2];
%求解变换矩阵 L
L = zeros(size(C));
for i = 1:m
    L(i,:) = C(i,:) * A - beta(i) * C(i,:);
end
%求解极点配置的状态反馈矩阵 F2
F2 = R * L;
%求解配置后的系统状态方程
A2 = A - B * F2;
B2 = B * R;
%配置极点后系统的传递函数矩阵
[numdp1,dendp1] = ss2tf(A2,B2,C,D,1);
[numdp2,dendp2] = ss2tf(A2,B2,C,D,2);
%配置极点后系统的阶跃响应曲线
t = 0:0.01:2;
step(A2,B2,C,D,1,t);
step(A2,B2,C,D,2,t);
```

图 5－16 是进行 a 阶极点配置后系统对第一个输入量的阶跃响应曲线。可以看出，在系统输出解耦的情况下实现了一阶极点配置，响应曲线满足要求。此时系统对第二个输入量的阶跃响应曲线见图 5－17。

可以求得系统此时的传递函数矩阵为

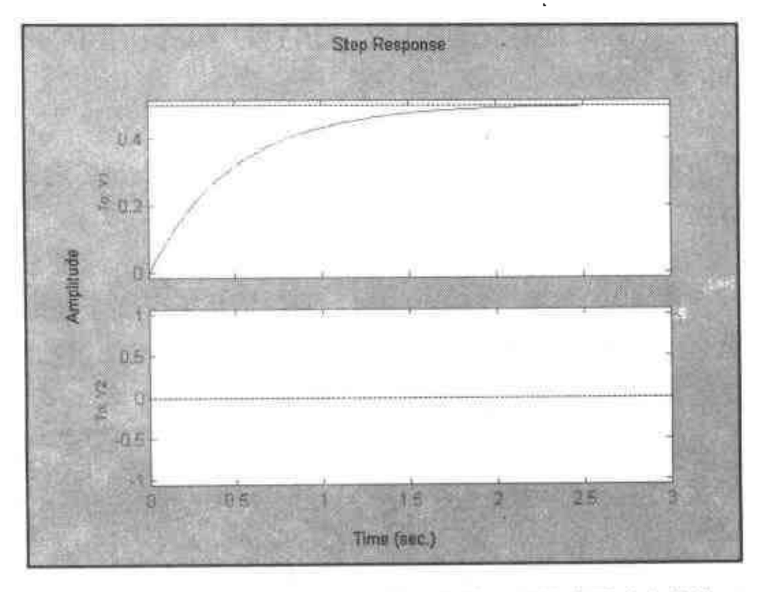


图 5-16　配置极点后系统对第一个输入量的阶跃响应曲线

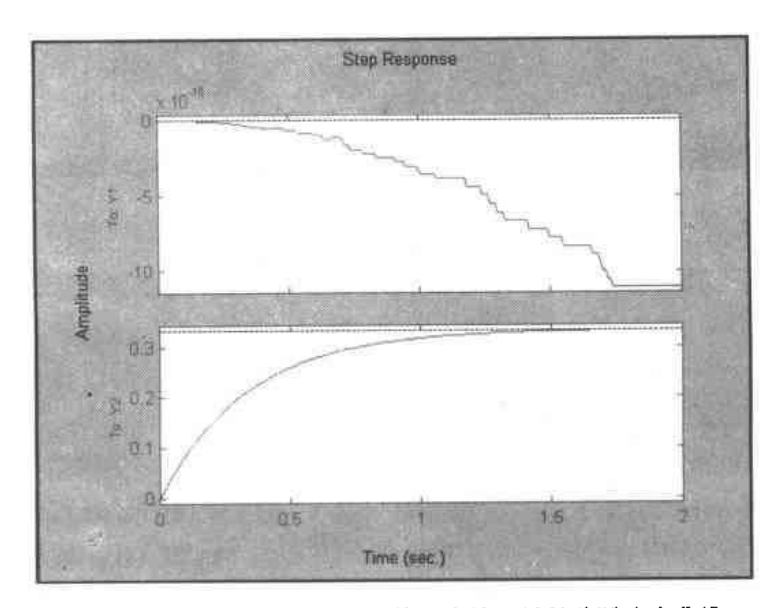


图 5-17　配置极点后系统对第二个输入量的阶跃响应曲线

```
numdp1 =    0      1        3        0
            0      0        0        0
dendp1 =    1      5        6        0
numdp2 =    0   -0.0000  -0.0000  -0.0000
            0    1.0000   2.0000   0
dendp2 =    1      5        6        0
```

即

$$G(s)=\frac{1}{s^3+5s^2+6s}\begin{bmatrix} s^2+3s & 0 \\ 0 & s^2+2s \end{bmatrix}=\begin{bmatrix} \frac{1}{s+2} & 0 \\ 0 & \frac{1}{s+3} \end{bmatrix}$$

可见，系统不但完全解耦，而且分别将两个解耦后的子系统的闭环极点配置在了 -2 和 -3。并且，图 5-16 和图 5-17 的响应曲线也符合要求，一方而实现了解耦控制，另一方而各子系统是稳定的。

有的读者可能会问，图 5-17 的第一个输出量对第二个输入量的响应曲线并不为零，是不是没有实现解耦控制？其实不是这样的，仔细看一下图 5-17 上半部分的纵坐标就会发现，这条曲线纵坐标的数量级是 10 的 -16 次方。可以肯定地说，这是由于 MATLAB 数值求解的误差造成的，当然，这种误差在工程领域是可以完全忽略不计的。并且，即使是这样小的误差，最终也会限制在很小的范围内。我们可以将横轴的时间坐标放大到 1000s，重新绘制这根曲线，有：

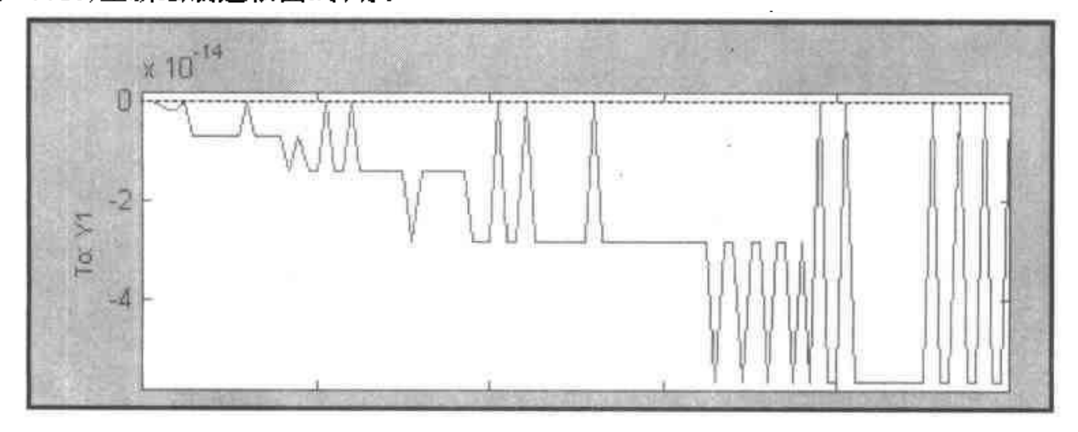


图 5-18 第一个输出量对第二个输入量的阶跃响应曲线

可见，响应误差最终稳定在 10 的 -14 次方数量级上，对最终解耦控制器的正常运行不会产生影响。

小结：本例比较详细地分析了多输入多输出系统的解耦控制器设计和极点配置的问题，并且给出了响应的 MATLAB 程序。

事实上，多变量系统的分析和求解还有许多其他的方法，例如多变量频域控制领域的对角优势法、逆 Nyquist 图法等等。因为都是依据矩阵运算和图形处理，所以 MATLAB 同样可以编写出这些算法的程序。本节并没有介绍新的 MATLAB 函数，从总体来看，到目前为止除了有关最优化的函数外，我们介绍的 MATLAB 函数已经足以胜任有关控制理论基础课程的学习和实践需要了。关键是掌握算法和思路，应用 MATLAB 这个得力的工具解决实际问题。

5.3 线性二次型最优控制器设计

对于线性系统，若取状态变量和控制变量的二次型函数的积分作为性能指标，这种动态系统最优化问题称为线性系统二次型性能指标的最优控制问题，简称线性二次型问题。它的最优解可以写成统一的解析表达式，而且可以导出一个简单的状态线性反馈控制律，其计算和工程实现都比较容易。

MATLAB 控制系统工具箱中提供了一些 LQ(Linear Quadratic，线性二次型)设计工具，可以很方便地完成线性二次型最优控制器的设计。下面我们简要介绍一下有关的理

论内容，然后给出一个应用实例。

5.3.1　代数 Riccati 方程求解

设线性系统的状态方程模型(A,B,C,D)已知，如果希望这样一个系统能够满足某种最优的要求，最简单的可以引入线性二次型最优控制指标，即：

$$J=\frac{1}{2}X^T(tf)SX(tf)+\frac{1}{2}\int_0^{tf}[X^T(t)Q(t)X(t)+u^T(t)R(t)u(t)]dt$$

其中 Q(t)和 R(t)分别是对状态变量和控制量的加权矩阵。一般情况下，假定这两个矩阵为定常矩阵(即不随时间变化)，并简记 Q(t)=Q，R(t)=R。线性二次型最优控制就是求出 J 最小时的控制量 u(t)，从而获得性能最优。为了达到这一目的，首先构造一个 Hamilton 函数：

$$H=-\frac{1}{2}[X^T(t)QX(t)+u^T(t)Ru(t)]+\lambda^T[AX(t)+Bu(t)]$$

然后通过求导的方法可以求出最优控制信号 u(t)为：

$$u(t)=-R^{-1}B^TP(t)X(t)$$

其中 P(t)矩阵就是 Riccati 方程的解：

$$\dot{P}(t)=-P(t)A-A^TP(t)+P(t)BR^{-1}B^TP(t)-Q$$

上面的方程是微分 Riccati 方程，一般是多个相互耦合的非线性微分方程组，除了特殊情况外，一般不存在解析解。这就给求解最优控制信号 u(t)造成了困难。因此，我们一般求解 u(t)的稳态解。即令 tf 趋于无穷，则 P(t)趋于一个常值矩阵，P(t)的一阶导数趋于零，有：

$$0=-PA-A^TP+PBR^{-1}B^TP-Q$$

上式被称为代数 Riccati 方程，其求解就比较容易了。并且，因为都是矩阵运算，用 MATLAB 实现起来也比较容易。并且，MATLAB 提供了一条求解代数 Riccati 方程的函数 care()，省去了我们手工编程的工作。其基本调用格式为：

```
[X,L,G,RR] = CARE(A,B,Q,R,S,E)
```

它所求解的代数 Riccati 方程形式为：

$$A^TXE+E^TXA-(E^TXB+S)R^{-1}(B^TXE+S^T)-Q=0$$

一般缺省设置 S=0，E=I，这样该方程就和原始的代数 Riccati 方程一致了。X 是求得的代数 Riccati 方程的解，L 是闭环状态方程参数矩阵的特征值，G 是状态反馈矩阵，RR 是残留矩阵的 Frobenius 范数。有了代数 Riccati 方程的解，我们就可以求出最优控制信号 u(t)和系统的响应曲线了。请看下例：

[例 5.4]　某控制系统的状态方程描述以及 Q、R 矩阵如下。试求解其代数 Riccati 方程的解和最优控制信号 u(t)，并绘制系统对两个输入量的阶跃响应曲线。

$$A=\begin{bmatrix}2&0&0&0\\0&-5&1&0\\0&0&-5&1\\0&0&0&-5\end{bmatrix}\quad B=\begin{bmatrix}2&0\\0&5\\0&3\\1&1\end{bmatrix}\quad C=\begin{bmatrix}1&1&0&0\\0&1&2&1\end{bmatrix}\quad D=\begin{bmatrix}0&0\\0&0\end{bmatrix}$$

$$Q=\begin{bmatrix}1&&&\\&2&&\\&&3&\\&&&4\end{bmatrix}\quad R=\begin{bmatrix}1&0\\0&1\end{bmatrix}$$

结果：系统的代数 Riccati 方程的解为

```
X =     0.2099     0.0018     0.0006    -0.0192
        0.0018     0.1549    -0.0270    -0.0215
        0.0006    -0.0270     0.2545     0.0048
       -0.0192    -0.0215     0.0048     0.3809
```

分析：本例是一个 2×2 的多输入多输出系统，因此最优控制信号是二维的。Q 是状态变量的加权矩阵，R 是控制信号的加权矩阵，其意义是衡量二者在性能指标 J 里的权重。除了要求它们必须是正定矩阵之外没有什么特殊要求。不过不同 Q 和 R 会得出不同的代数 Riccati 方程的解，进而会得出不同的最优控制信号 u(t)。下一节我们将具体讨论 Q 和 R 的选取规则。

求解过程：

本例的解题步骤分为以下几步：

1．求解代数 Riccati 方程

调用 care()函数可以很快得到代数 Riccati 方程的解。为了比较详细的阐述线性二次型最优控制器的求解步骤，我们调用简单格式的 care()函数。在 MATLAB Editor/Debugger 下输入以下代码：

```
%清除所有内存变量
clear all;
%系统的状态方程
A =    [-2    0    0    0;
        0    -5    1    0;
        0     0   -5    1;
        0     0    0   -5];
B =    [2     0;
        0     5;
        0     3;
        1     1];
C =    [1     1    0    0;
        0     1    2    1];
D = zeros(2);
%Riccati 方程的 Q、R 矩阵
Q = diag(1:4);
R = eye(2);
%求解代数 Riccati 方程的解
X = care(A,B,Q,R);
```

求得的本系统的代数 Riccati 方程的解请参看上一部分“结果”中的数据。

2．求解系统的最优控制信号

根据最优控制信号的表达式：

$$u(t) = -R^{-1}B^{T}PX(t)$$

得到系统的代数 Riccati 方程的解之后可以求出最优控制信号的变化曲线，其中，状态反馈矩阵 K 等于：

$$K = -R^{-1}B^{T}P$$

紧接上一步，输入以下代码：

```
%系统的状态反馈矩阵
K = -inv(R)*B'*X;
%仿真时间
t = 0:0.005:0.5;
%系统的状态方程抽象描述
sys = ss(A + B*K,B,C,D);
%求解系统的状态变量 x(t)
[y,T,xt] = step(sys);
%求解最优控制信号 u(t)
n = length(T);
m = length(R);
for i = 1:m
    for j = 1:n
        u(j,:,i) = K(i,:)*(xt(j,:,i))';
    end
end
%图形绘制
subplot(211),
%最优控制信号 u(1)
plot(T,u(:,:,1))
title('Linear Quadratic Controller output u1(t)')
xlabel('Time - sec')
ylabel('Value');
grid;
subplot(212),
%最优控制信号 u(2)
plot(T,u(:,:,2))
title('Linear Quadratic Controller output u2(t)')
xlabel('Time - sec')
ylabel('Value');
grid;
```

得到系统的二维最优控制信号如图 5－19 所示。

从图 5－19 中可以看出，在阶跃输入下，控制信号基本上在 0.5～0.7s 左右就趋于稳定，也就是说，系统的过渡过程时间不会大于这个值。当然，在实际的控制系统中，最优信号的值是不必手工计算的，只需求出状态反馈矩阵，也就是最优控制器即可。这里求解出系统的最优控制信号变化曲线，一方面是帮助读者理清最优控制器的设计思路，另一方面

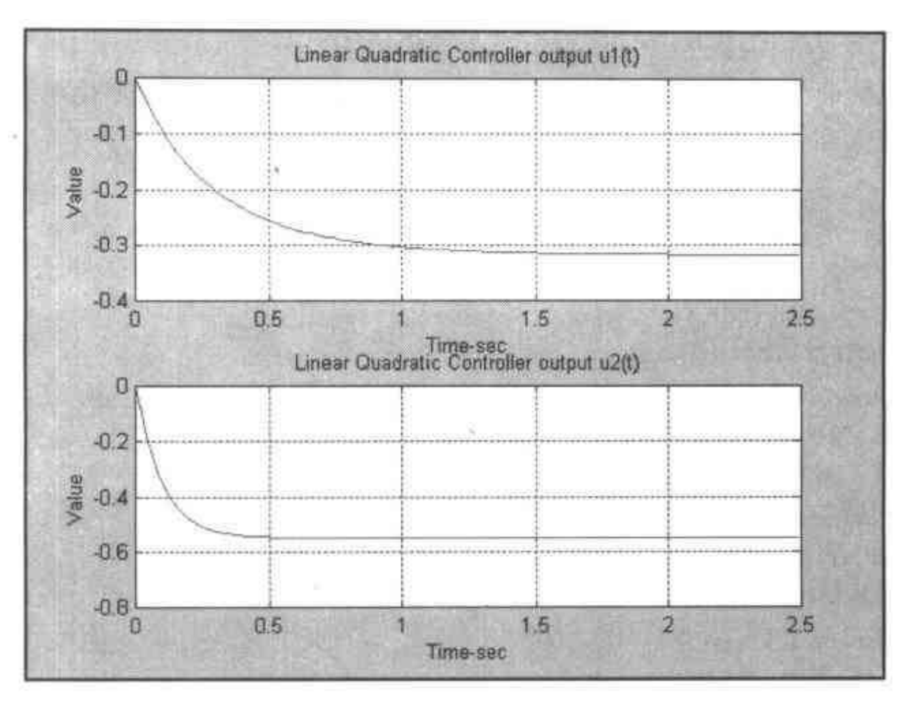


图 5-19　最优控制信号变化曲线

也是系统运动状态的一种反映。

3. 系统对输入的响应曲线

紧接上一步，在 MATLAB Command Window 键入下列语句即可获得图 5-20 和图 5-21所示的曲线：

```
step(A + B * K,B,C,D,1,t)
step(A + B * K,B,C,D,1,t)
```

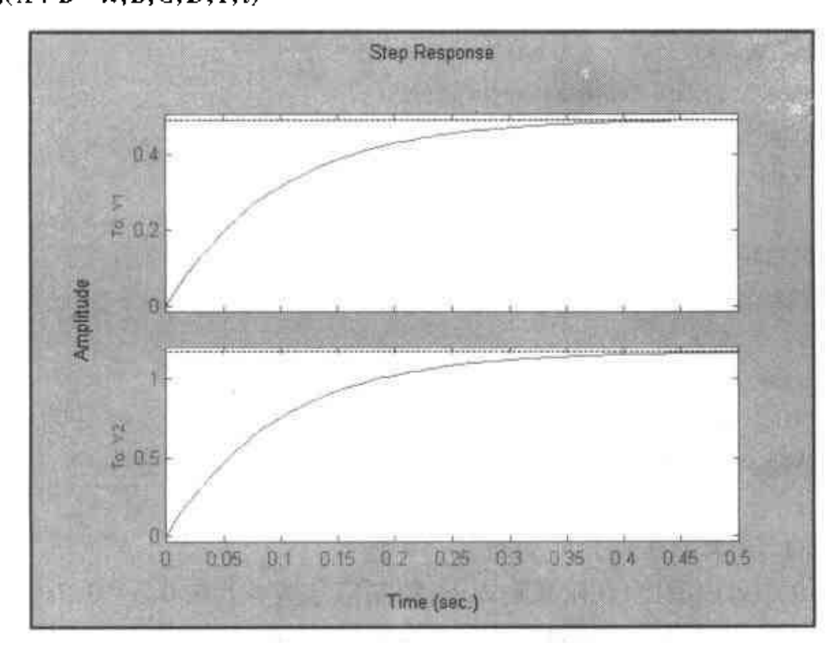


图 5-20　系统对第一个输入量的阶跃响应曲线

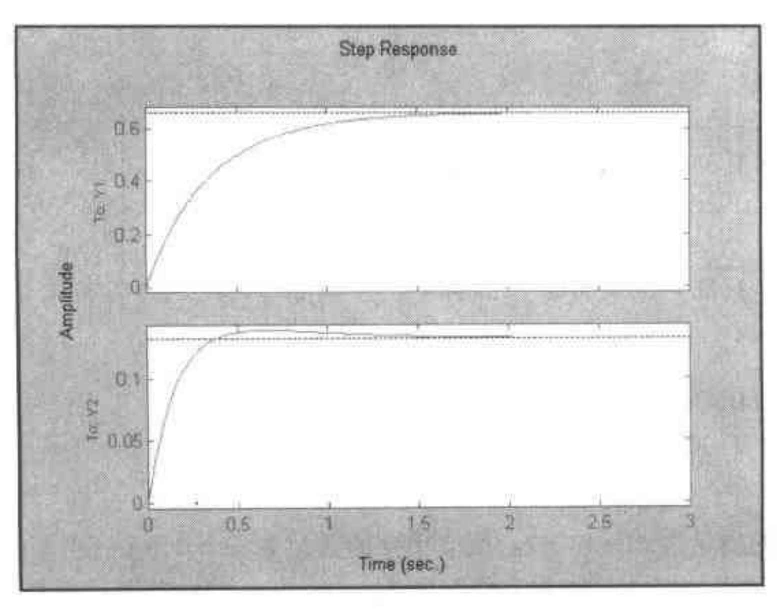


图 5-21　系统对第二个输入量的阶跃响应曲线

这里求解系统的闭环参数矩阵时用的是 A+B*K 而不是 A-B*K，这是因为在计算状态反馈矩阵 K 时已经包含负号了。从图 5-20 和图 5-21 中还可以看出，系统的过渡过程时间都在 0.5～0.7s 之间，这也验证了第二步的说法。

小结：本例的重点是使用 MATLAB 求解代数 Riccati 方程的解及其在求解最优控制信号中的应用，因此没有对系统进行解耦控制和性能指标的设计。有关这方面的内容，我们将在下一小节讨论。

5.3.2　线性二次型最优控制器设计举例

使用线性二次型最优控制器进行控制系统设计和校正的最大优点就是不必根据要求的性能指标确定闭环极点的位置，只需根据系统的响应曲线寻找出合适的状态变量和控制量的加权矩阵即可。因为求得的控制器是误差指标 J 最优意义下的控制器，所以系统的性能也是 J 指标意义下最优的。

[例 5.5]　某倒立单摆系统如图 5-22 所示，其状态方程已知。

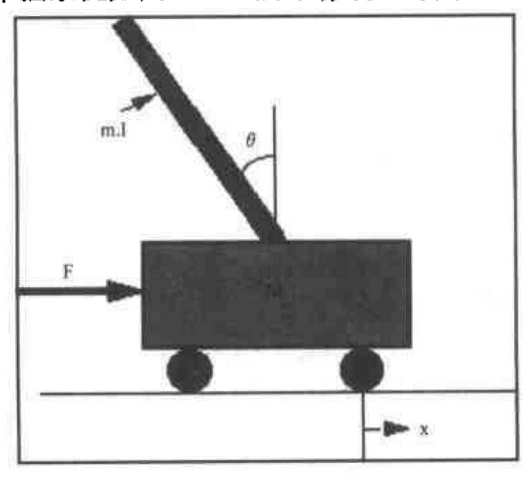


图 5-22　倒立单摆系统示意图

$$\begin{bmatrix}\dot{x}\\ \ddot{x}\\ \dot{\Phi}\\ \ddot{\Phi}\end{bmatrix}=\begin{bmatrix}0 & 1 & 0 & 0\\ 0 & \dfrac{-(I+ml^2)b}{I(M+m)+Mml^2} & \dfrac{m^2gl^2}{I(M+m)+Mml^2} & 0\\ 0 & 0 & 0 & 1\\ 0 & \dfrac{-mlb}{I(M+m)+Mml^2} & \dfrac{mgl(M+m)}{I(M+m)+Mml^2} & 0\end{bmatrix}\begin{bmatrix}x\\ \dot{x}\\ \Phi\\ \dot{\Phi}\end{bmatrix}+\begin{bmatrix}0\\ \dfrac{I+ml^2}{I(M+m)+Mml^2}\\ 0\\ \dfrac{ml}{I(M+m)+Mml^2}\end{bmatrix}u$$

$$y=\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 0 & 1 & 0\end{bmatrix}\begin{bmatrix}x\\ \dot{x}\\ \Phi\\ \dot{\Phi}\end{bmatrix}+\begin{bmatrix}0\\ 0\end{bmatrix}u$$

其中小车的质量为 $M=0.5$ kg，倒立单摆的质量为 $m=0.5$ kg，小车的摩擦系数为 $b=0.1$ N/m/s，端点与倒立单摆质心的距离为 $l=0.3$ m，倒立单摆的惯量为 $I=0.006$ kg $*$ m^2，输入量 $u=F$ 是施加在小车上的外力，四个状态变量分别是小车的坐标 x、$\dot{x}$ 的一阶导数，倒立单摆的垂直角度 Φ，$\dot{\Phi}$ 的一阶导数。输出的被控量分别是小车的坐标 x 和倒立单摆的垂直角度 Φ。试根据误差指标 J 最优意义下最优的规则设计线性二次型最优控制器和相关的参考输入以及观测器，满足以下指标：

1. 输出量 x 和 Φ 的过渡过程时间小于 2s。
2. 输出量 x 的上升时间小于 0.5s。
3. 输出量 Φ 的超调量小于 20° (0.35 rad/s)。

结果：求得的线性二次型最优控制器为：

K = -70.7107 -37.8345 105.5298 20.9238

分析：本例是一个单输入双输出系统，采用属于经典控制方法的 PID 校正和根轨迹校正都无法满足同时控制两个输出量的要求。如果采用状态空间的极点配置方法，必须分别设计两个反馈校正装置，根据不同的性能指标分别配置极点 而采用 LQ 方法就不必这样繁琐，只需设计一个状态反馈校正装置，在性能指标 J 最小的意义下求得最优控制信号即可。MATLAB 还提供了一条直接求解 LQ 最优状态反馈的函数 lqr()，我们可以不必根据 care()函数求解的数据进行计算。其基本调用格式为：

[K,S,E] = LQR(A,B,Q,R)

其中 A，B，Q，R 的意义如前所述，K 是求得的最优状态反馈矩阵，S 是相应的代数 Riccati 方程的解，E 是闭环系统的特征值。

求解过程：

本例的解题步骤分为以下几步：

1. 分析原系统的开环阶跃响应

首先要求得其开环系统的特征值，判断其稳定性。然后根据其阶跃响应曲线分析当前的运动情况与期望性能指标之间的差距，确定校正的手段。在 MATLAB Editor/Debugger 下输入以下代码：

```
%清除所有内存变量
clear all;
```

```
%系统参数初始化
M = 0.5;
    m = 0.2;
    b = 0.1;
    i = 0.006;
    g = 9.8;
    l = 0.3;
%系统的状态方程描述
    p = i*(M+m)+M*m*l^2;
    A =     [0              1                  0          0;
             0    -(i+m*l^2)*b/p(m^2*g*l^2)/p             0;
             0              0                  0          1;
             0    -(m*l*b)/p      m*g*l*(M+m)/p 0];
    B =   [0;  (i+m*l^2)/p;  0;  m*l/p];
    C =   [1    0    0    0;
           0    0    1    0];
    D =   [0;0];
    %求解系统的特征值
    p = eig(A);
    t=0:0.01:1;
    step(A,B,C,D,1,t)
```

求得系统的特征值为：

```
p =    0
    -0.1428
     5.5651
    -5.6041
```

系统有一个右半平面的极点，因此不稳定，必须加入校正装置。此时系统的阶跃响应曲线如图 5－23 所示。

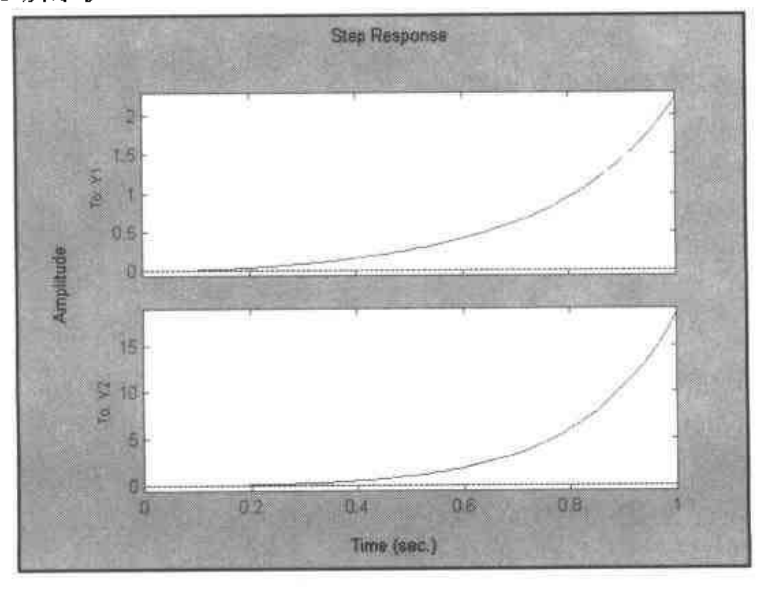


图 5－23　原系统的阶跃响应曲线

图 5-23 上半部分是小车坐标 x 的阶跃响应曲线，下半部分是倒立单摆的垂直角度 Φ 的阶跃响应曲线。从图中可以看出它们都不稳定，谈不上与期望的性能指标作比较。下面加入 LQ 校正装置。

2. 线性二次型最优控制器设计

设计线性二次型最优控制器关键是选择加权矩阵 Q。一般说来，Q 选择的越大，系统达到稳态所需的时间越短，当然，还要实际的系统允许。我们首先选择 Q = C' * C，R = I，然后根据实际情况进行调节。输入以下代码：

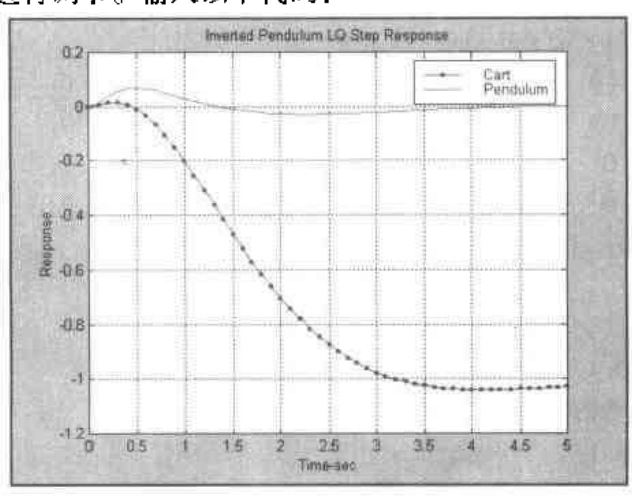


图 5-24 初步 LQ 校正后系统的阶跃响应曲线

```
%Q 和 R 矩阵的选择
x = 1;
y = 1;
Q =   [x     0     0     0;
       0     0     0     0;
       0     0     y     0;
       0     0     0     0];
R = 1;
%求解线性二次型最优状态反馈矩阵
K = lqr(A,B,Q,R)
%求解系统闭环状态方程
Ac = [(A - B * K)];
Bc = [B];
Cc = [C];
Dc = [D];
%输出系统阶跃仿真
T = 0:0.02:2;
U = ones(size(T));
[Y,X] = lsim(Ac,Bc,Cc,Dc,U,T);
plot(T,Y(:,1),'.-',T,Y(:,2));
title('Inverted Pendulum LQ Step Response');
```

```
xlabel('Time-sec');
Ylabel('Response');
grid;
legend('Cart','Pendulum');
```

此时求得的线性二次型最优状态反馈矩阵为:

K = -1.0000 -1.6567 18.6854 3.4594

从图 5-24 系统的阶跃响应曲线中可以看到,系统超调量基本满足要求,但一方面系统的稳态值与期望相差很远(小车坐标的响应曲线稳态值为负值),另一方面过渡过程时间和上升时间都很大,必须重新校正。方法就是加大加权矩阵 Q 的值。经不断仿真发现,当 x=5000,y=100 时效果比较理想。因此,在 MATLAB Editor/Debugger 下将上一段代码中的 x 和 y 改成相应的值,重新运行代码,有:

求得的线性二次型最优状态反馈矩阵为:

K = -70.7107 -37.8345 105.5298 20.9238

可以看到,该状反馈矩阵态要明显大于上一步设计的反馈矩阵。此时系统的阶跃响应曲线如图 5-25 所示。

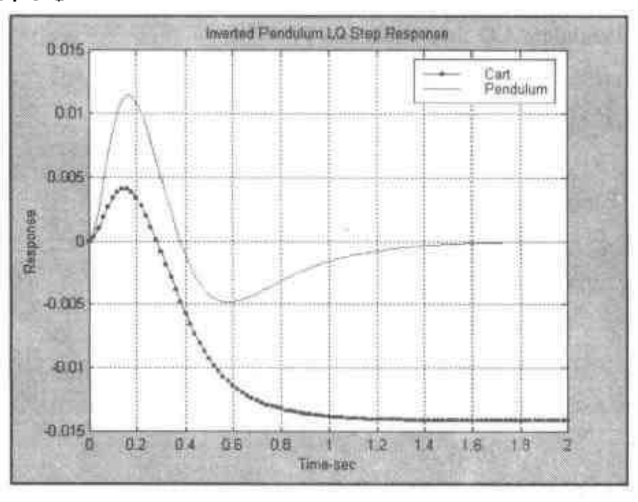


图 5-25 加入 LQ 校正后系统的阶跃响应曲线

从图 5-25 中可以看到,系统响应的快速性得到了明显的改善,上升时间和过渡过程时间都满足最初设计的要求,但稳态值依旧没有得到改善。这是因为没有加入参考输入的原因。因此,为了保持解题的完整性,我们下一步将设计系统的参考输入和状态观测器,使得系统的响应完全符合要求的性能指标。

3. 参考输入及状态观测器设计

参考输入和状态观测器的功能和设计方法在前文已经有详细的解释,这里就不再赘述了。一般说来,经过线性二次型最优化方法设计的控制器投入使用前都要设计参考输入和状态观测器,以满足实际需要。紧接上一步,输入以下代码:

```
%参考输入倍数
Cn=[1 0 0 0];
Nbar=rscale(A,B,Cn,0,K)
Bcn=[Nbar*B];
```

```
%观测器极点
P = [-40  -41  -42  -43];
%观测器状态反馈矩阵
L = place(A',C',P)'
%加入观测器和参考输入后系统的状态方程描述
Ace = [A-B*K          B*K;
    zeros(size(A)) (A-L*C)];
Bce = [         B*Nbar;
    zeros(size(B))];
Cce = [Cc zeros(size(Cc))];
Dce = [0;0];
%阶跃响应曲线仿真
T = 0:0.02:2;
U = ones(size(T));
[Y,X]=lsim(Ace,Bce,Cce,Dce,U,T);
plot(T,Y(:,1),'.-',T,Y(:,2));
title('Inverted Pendulum LQ Step Response');
xlabel('Time-sec');
Ylabel('Response');
grid;
legend('Cart','Pendulum');
```

对应的观测器状态反馈矩阵为:

```
L =   1.0e + 003  *
       0.0826        -0.0010
       1.6992        -0.0402
      -0.0014         0.0832
      -0.0762         1.7604
```

图 5-26 就是最终的系统阶跃响应曲线,两条曲线的过渡过程时间都小于 2s,小车坐标 x 的上升时间小于 0.5s,倒立单摆的垂直角度 Φ 的第一个正的峰值也小于 0.35rad/s。

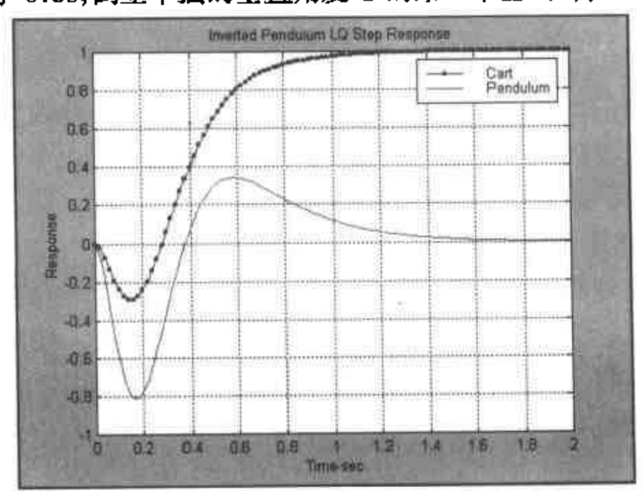


图 5-26　加入参考输入和状态观测器后系统的阶跃响应曲线

我们还可以绘制出系统的最优控制信号：

相应的代码为：

```
K = [K,0,0,0,0];
n = length(T);
    for j = 1:n
        u(j,:) = K * (X(j,:))';
    end
plot(T,u);
title('Inverted Pendulum LQ Controllor output:u(t)');
xlabel('Time - sec');
Ylabel('Response');
grid;
```

从图 5 - 27 中看到，控制信号最大输出峰值接近 - 100，实际应用中尤其是在工业控制领域要考虑控制器的阀门的量程限额。不过我们现在仅仅是从理论上阐明系统的运动和校正手段，就不再讨论这个问题了。

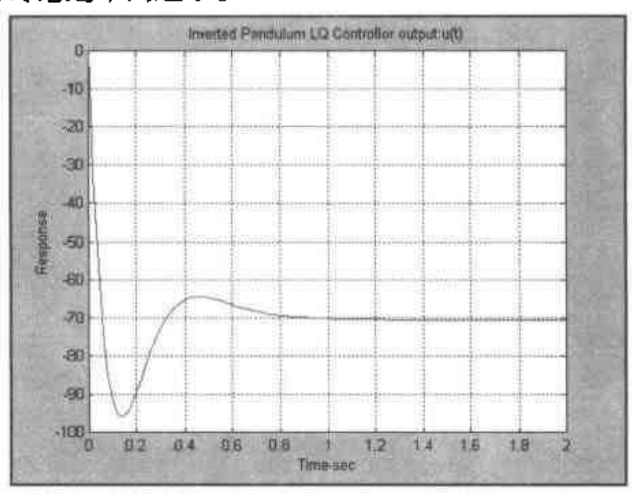


图 5 - 27　倒立单摆系统的线性二次型最优控制信号

小结：本例详细介绍了有关线性二次型最优控制器的设计方法和 MATLAB 实现手段，重点是 lqr()函数的应用。事实上，使用 care()函数也可以进行线性二次型最优控制器的设计。不过 lqr()函数的缺省返回值是状态反馈矩阵 K，而 care()函数的缺省返回值是代数 Riccati 方程的解 X。本例还使用了一个新的函数 legend()，其功能是为 plot()函数绘制的图形增加注释。该函数能够自动识别不同的曲线颜色和绘制的线段类型。其具体参数设置和使用方法请参看 MATLAB 帮助。

5.4　小　　结

本章主要介绍了控制系统各种常用的状态空间设计方法在 MATLAB 下的实现手段，包括通过状态反馈进行极点配置、状态观测器设计、状态空间的解耦控制、代数 Riccati 方程求解、线形二次型最优控制器设计等等。其中最基础也是最常用的是状态反馈进行

极点配置和状态观测器设计，其基本思路和实现方法就是控制系统状态空间设计的基本方法。这方面MATLAB提供了功能强大的函数，可供我们使用。

到此为止，有关控制理论的基本概念和方法及其在MATLAB中的实现和应用就基本介绍完了。从下一章开始，我们转入MATLAB下的SIMULINK仿真环境的介绍。

第六章　SIMULINK 进阶

SIMULINK 自 1992 年问世以来，很快在控制界有了广泛应用。它的前身是 1990 年 MathWorks 公司为 MATLAB 提供的控制系统模型图形输入和仿真工具 SIMULAB。MATLAB5.2 提供的 SIMULINK 版本为 2.0，本书的讲述也以此版本为基础。

概括地说，SIMULINK 是一个可视化动态系统仿真环境。一方面，它是 MATLAB 的扩展，保留了所有 MATLAB 的函数和特性；另一方面，它又有可视化仿真和编程的特点。第四章里我们曾经介绍过 MATLAB 下的可视化根轨迹校正工具 rltool，它们的内涵有些相似。不过使用方法大不相同，并且 SIMULINK 的功能要强大得多。借助少量的 MATLAB 函数，我们在前五章介绍的内容基本上都可以在 SIMULINK 下完成。并且设计更直观，求解更方便。

借助其可视化的优点，使用 SIMULINK 可以分析非常复杂的控制系统。一般说来，SIMULINK 的功能有两部分，其一是系统建模，其二是系统分析。当然，对于控制系统的设计者来说，这两部分是一个连贯的整体。但是从解决问题的方法上来说，还是有区别的。因为建立好模型之后，可以在 SIMULINK 环境下直接分析，也可以在 MATLAB Command Window 下使用 MATLAB 函数进行分析。当然，结果是相同的。

在 SIMULINK 环境下特别适用于相同的方框图模型，可以直接建立相同的方框图进行仿真。并且，SIMULINK 的命令基本上都是鼠标驱动的(mouse - driven commands)。仿真的结果可以在 SIMULINK 下观看，也可以在 MATLAB Workspace 中观看。

本章是 SIMULINK 的初步介绍，目的是使读者熟悉 SIMULINK 环境及其基本使用方法。对 SIMULINK 比较熟悉的读者可以直接进入下一章。

6.1　SIMULINK 与控制系统

事实上，自从 SIMULINK 问世以来，很大一部分控制系统的设计和仿真问题都是在 SIMULINK 环境下完成的。在 SIMULINK 环境下分析和设计控制器，甚至不需要编制任何程序就可以得到希望的参数。并且，SIMULINK 秉承了 MATLAB 的优点，所有仿真结果都可以通过图形来显示。

我们首先从一个具体的例子入手，让没有接触过 SIMULINK 的读者先有一个感性的认识，然后再具体介绍 SIMULINK 的使用方法。

6.1.1　SIMULINK 下控制系统仿真举例

如前所述，在 SIMULINK 下进行控制系统仿真和设计分为两步，其一是系统建模，其二是系统分析和仿真。请看下面这个例子：

［例 6.1］　某控制系统传递函数如下。试在 SIMULINK 环境下构建系统的方框图并对系统的阶跃响应进行仿真。

$$G(s)=\frac{s+3}{s^2+4s+8}\cdot\frac{1}{s+3}$$

结果：在 SIMULINK 环境下求得系统的响应曲线如图 6－1 所示。

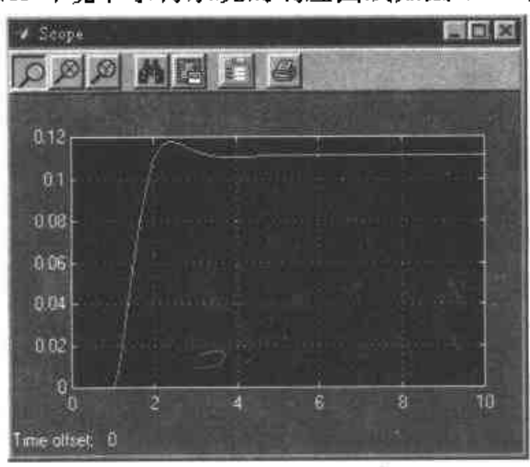


图 6－1　系统阶跃响应曲线

求解过程：

本例的求解分为以下几步：

1．进入 SIMULINK 环境



图 6－2　从 MATLAB 进入 SIMULINK 环境

在 MATLAB Command Window 下键入“simulink”，即可进入如图 6－3 所示的 SIMULINK 仿真环境和如图 6－4 所示的仿真元件库。

该仿真环境上边是选择菜单，包括“File”、“Edit”、“Help”等 Windows 常用菜单和“View”、“Simulation”、“Format”等 SIMULINK 独有的菜单项。选择菜单下边是工具栏，包括“新建”、“存储”、“运行”等常用命令的快捷图标。中间大片的空白部分是仿真环境的主体，系统方框图的搭建就在这里进行。最下边一行是状态栏，显示仿真环境当前的状态以及仿真算法。

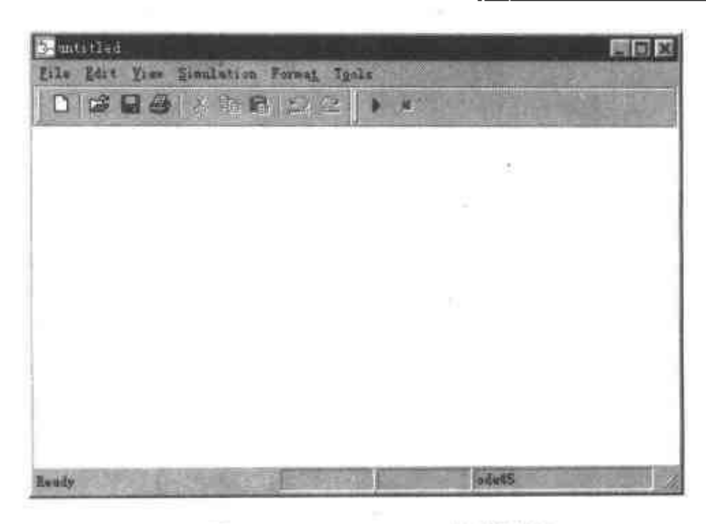


图 6-3 SIMULINK 仿真环境

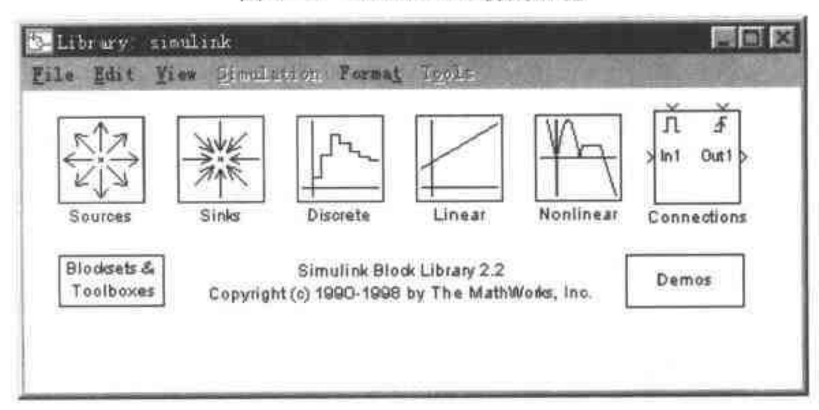


图 6-4 SIMULINK 元件库

系统的方框图就是通过选择图 6-4 元件库中的元件来搭建的。

从功能上讲，如图 6-3 所示的 SIMULINK 仿真环境相当于 MATLAB Command Window，模型的构建和仿真都要在这个主窗口进行。而图 6-4 所示的仿真元件库就相当于 MATLAB 环境下的各种工具箱以及内置的函数，通过对这些元件库里元件的选取完成模型的搭建。各元件库的具体作用将在下一节介绍。

2. 进入"Linear"元件库，构建传递函数

用鼠标双击图 6-4 界面下标有"Linear"的图标 ，进入如图 6-5 所示的线性系统元件库。

在 SIMULINK 环境下读者见到的线性系统元件库界面是长方形的，这里为了节省空间起见将其长度缩短了。在该界面下用鼠标双击标有"Transfer Fcn"的图标，进入图6-6的界面设置系统的传递函数。

线性系统元件库里包含了我们在第二章里讲述的绝大部分控制系统的描述形式。我们选择的"Transfer Fcn"图标是系统的传递函数描述。因为原系统的开环传递函数由两个分式相乘构成，所以我们要用鼠标将"Transfer Fcn"图标拖到 SIMULINK 仿真环境下两次，当然，用"Ctlr-C"、"Ctlr-V"操作也可以完成。拖到仿真环境下就形成了两个传递函数块，双击此模块，进入图 6-6 的界面。

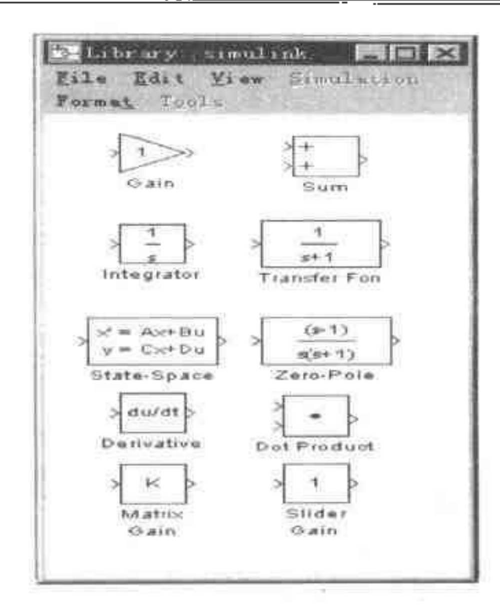


图 6－5 线性系统元件库

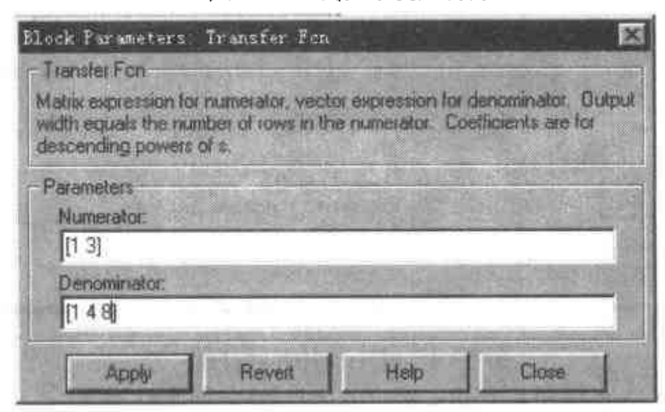


图 6－6 设置系统的传递函数

在图 6－6 界面下我们可以设置传递函数的系数，其表达方式和 MATLAB 环境下相同："Numerator"选项是传递函数分子多项式系数的降幂排列，"Denominator"选项是分母多项式系数的降幂排列。我们将原系统开环传递函数的两个分式的系数分别填入这两个传递函数块。

3．进入"Source"元件库，选取阶跃函数

用鼠标双击图 6－4SIMULINK 元件库界面下标有"Source"的图标，进入输入源元件库如图 6－7 所示。

该输入源元件库包括控制系统设计和仿真领域常用的各种信号源和函数发生器。包括我们在前几章里经常用到的阶跃函数发生器、正弦函数发生器、脉冲函数发生器以及随机数发生器等等。

在图 6－7 所示的界面下选择标有"Step"的图标，用鼠标拖到如图 6－3 所示的 SIMULINK 仿真环境下，形成一个阶跃函数输入块。需要注意的是，Step 块缺省的起始时间是第 1 秒而非第 0 秒。双击该 Step 块即可对仿真起始时间和阶跃值的大小进行设置，不过对我们的仿真影响不大，这里就不再设置了。

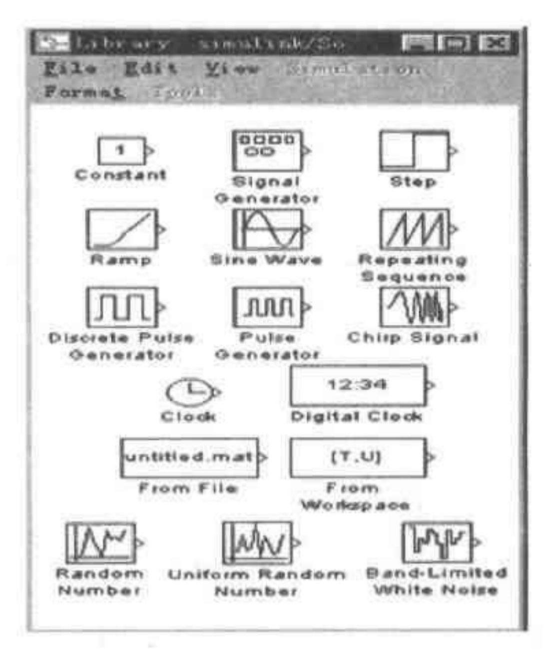


图 6－7　输入源元件库

4．进入“Sinks”输出方式元件库，选择输出方式

用鼠标双击图 6－4SIMULINK 元件库界面下标有“Sinks”的图标 ，进入输出方式元件库：

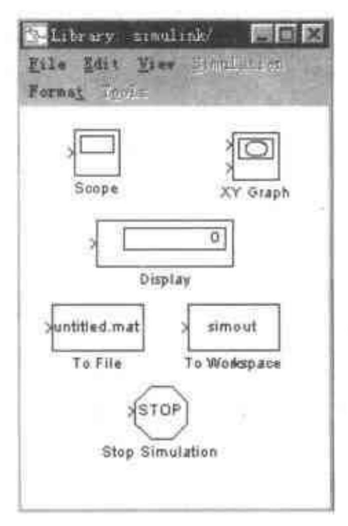


图 6－8　输出方式元件库

该元件库中包括连续、离散示波器以及输出到文件和输出到工作空间等各种输出方式。这里我们只选择标有“Scope”的示波器图标 ，将其拖到仿真环境下，方法如前所述。该元件的参数也可以设置，不过本例只使用缺省参数。

此外，为了形成闭环负反馈，还要拖住图 6－5 线性系统元件库界面下标有“Sum”的图标 ，拖到仿真环境下，并用鼠标双击将其设置为负反馈形式“＋ －”。

设置完毕的 Sum 块变为 形式。这样一来我们就一端接受输入的阶跃信号，一端

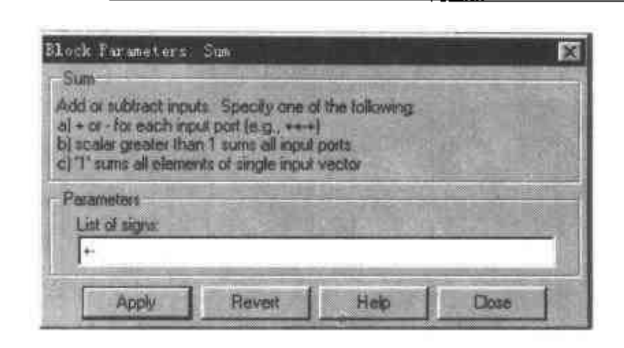


图 6-9 设置 Sum 块

接受输出的反馈信号，形成负反馈了。

5. 连接各元件，构成闭环传递函数并仿真

现在我们回到如图 6-3 所示的 SIMULINK 仿真环境，用鼠标划线，将各个元件连接成一个完整的方框图，如图 6-10 所示。

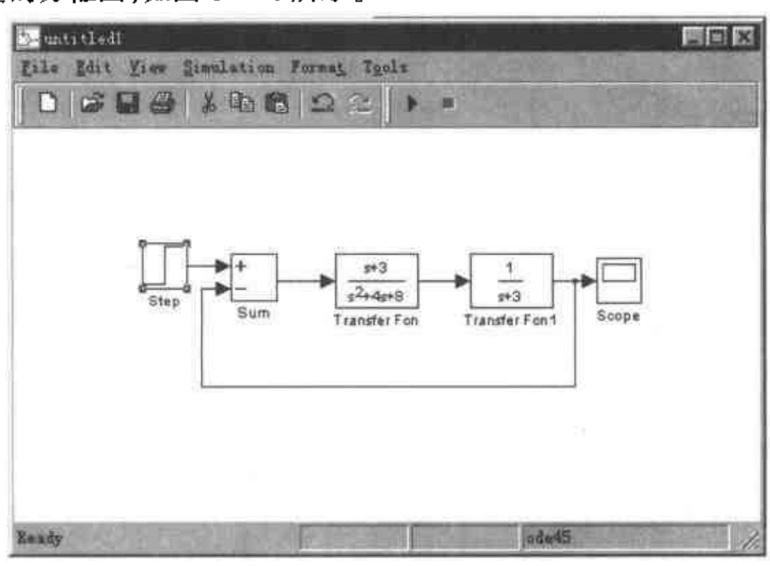


图 6-10 SIMULINK 仿真环境下构建的系统方框图

搭建好系统的方框图之后，就可以进行系统仿真了。用鼠标单击图 6-10 界面的“Start/pause Simulation” 按钮，SIMULINK 便自动运行仿真环境下的系统方框图模型。不过运行完之后还不能直接看到结果，必须用鼠标双击“Scope” 元件，这样就可以看到如图 6-1 所示的系统阶跃响应曲线了。开始看到的图形的坐标比例可能并不适当，和第四章介绍的 rltool 可视化界面一样，我们可以通过该图第二行的工具条来调节横坐标和纵坐标的比例，不过一般是用鼠标单击“Full View” 按钮，从而获得如图 6-1 所示的系统响应曲线的全景图。

有了系统的仿真曲线之后就可以分析系统的性能了。并且，根据不同的设计要求，我们可以随意在系统的方框图内添加新的控制器或校正环节，只需用鼠标改变各环节之间的连接即可。这比在 MATLAB Command Window 下或 MATLAB Editor/Debugger 下修

改传递函数系数要方便和直观的多了。

搭建系统方框图完成之后还可以存储起来以备下次使用。具体的方法和在 Windows 环境下存储一般的文件没有什么区别:在图 6-3 所示的 SIMULINK 仿真环境界面下选择"File"菜单的"Save"选项,设置文件名和路径即可。不过与以前编写的 M 文件不同,SIMULINK 模型文件的后缀是".mdl"

小结:本例介绍了一个简单的 SIMULINK 下三阶系统仿真的例子。从求解过程中我们可以看出,在 SIMULINK 环境下构建的系统模型和平常使用的控制系统方框图非常相似,系统的结构和各环节之间的关系一目了然。

本例的系统非常简单,手工编制 MATLAB 程序求解也不是很困难,因此还不能完全体现 SIMULINK 在系统建模方面的优势。对于那些有几十上百个环节的控制系统,这种可视化的建模方法就比单纯在文字界面下输入方便了不知多少倍。回想我们在第三章介绍的用关联矩阵的方法输入系统的方框图,一方面要求解关联矩阵,作大量的手工计算,另一方面还不能保证准确性。而在 SIMULINK 环境下,所有的模型搭建和仿真都是可视化操作。尤其是对于不习惯编程处理控制系统仿真的读者,甚至不用编写一行代码即可完成相当复杂的控制系统的模型构建和仿真。

本节希望通过一个简单的例子使读者对 SIMULINK 环境有一个感性的认识,下一节将详细介绍 SIMULINK 环境的功能和使用。

6.1.2 SIMULINK 环境

通过对例 6.1 的分析和求解,相信读者对 SIMULINK 环境已经有了初步的认识。下面系统的介绍 SIMULINK 的各项功能。

1. 进入 SIMULINK 环境

从 MATLAB 环境下进入 SIMULINK 一般有三种方式:

(1) 第一种是如例 6.1 所述的在 MATLAB Command Window 下键入"simulink";

(2) 第二种是选择 MATLAB Command Window 下"File"菜单的"New"-"Model"选项;

(3) 第三种是用鼠标单击 MATLAB Command Window 工具条中标有"New Simulink Model"的 ■ 按钮。

无论选择哪种方式,都会进入 SIMULINK 仿真环境和如图 6-11 所示的元件库。

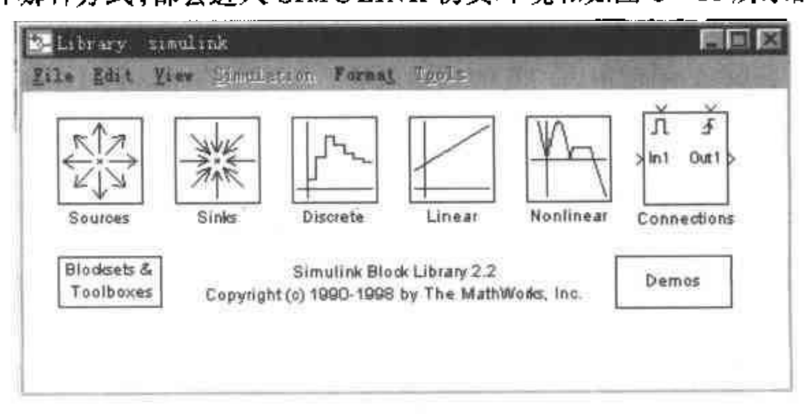


图 6-11 SIMULINK 元件库

2. SIMULINK 元件库

如图 6 – 11 所示的元件库里的内容就是 SIMULINK 的主体部分。无论是线性系统还是非线性系统,无论是系统建模还是系统仿真,主要都是使用该元件库提供的各种元件和模块。因此,我们将详细介绍此元件库的各项功能。

上一小节里我们已经介绍过其中的三个元件库,分别是输入源元件库、输出方式元件库和线性系统元件库,如图 6 – 12 所示。

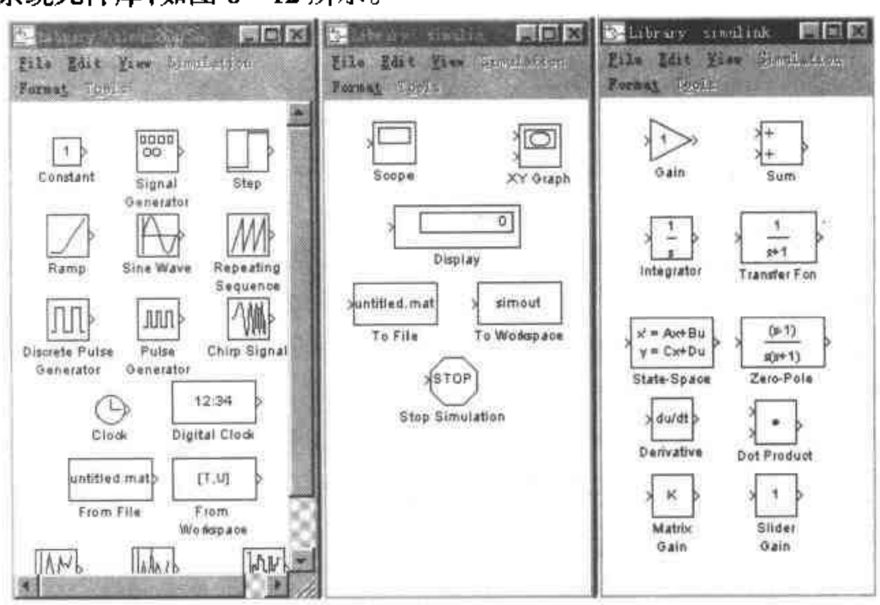


图 6 – 12 输入源、输出方式和线性系统元件库

输入源元件库里除了例 6.1 使用过的“Step”阶跃函数元件外,常用的还有:

(1) 标有“Siganal Generator”的 ,信号发生器。该发生器可以产生给定频率和幅值的正弦波(sine wave)、方波(square wave)和锯齿波(sawtooth wave),双击该图标即可进行设置。

(2) 标有“Clock”的 ,定时器。其作用是显示仿真时间。系统仿真时打开该元件,即可看到实时的仿真时间,对于定时仿真的系统来说,这个功能比较有用。不过这样会增加仿真的总的耗时。

(3) 标有“From Workspace”的 ,从工作空间输入。该元件的功能是从 MATLAB Workspace 输入已有的函数作为仿真的激励信号。首先要在 MATLAB 环境下建立其一个时间向量和相应的函数值向量,然后双击该图标将时间向量和函数值向量的名称填入指定的对话框即可。

输出方式元件库里除了例 6.1 中使用的“Scope”元件外,常用的还有标有“To Workspace”的 ,输出到工作空间。它的功能和“From Workspace”正好相反,是把仿真的结果连同输入信号输出到工作空间中去。

线性系统元件库里除了例 6.1 中使用的“Sum”元件和“Transfer Fcn”外,常用的还有:

(1) 标有"Gain"的 ,增益元件。其功能是改变线性系统的增益。双击该图标可以设置增益的大小。

(2) 标有"State－Space"的 ,状态空间模型元件。通过该元件可以将系统的状态空间模型与方框图结合起来。其参数设置格式也是矩阵的形式,与 MATLAB 下完全相同。

(3) 标有"du/dt"的 ,微分元件。通过该元件可以在系统中设置微分环节,尤其是在 PID 校正装置设计过程中,非常方便。

图 6－11 中第一行还有其他三个元件库:离散系统元件库、非线性系统元件库和连接装置元件库,如图 6－13 所示。

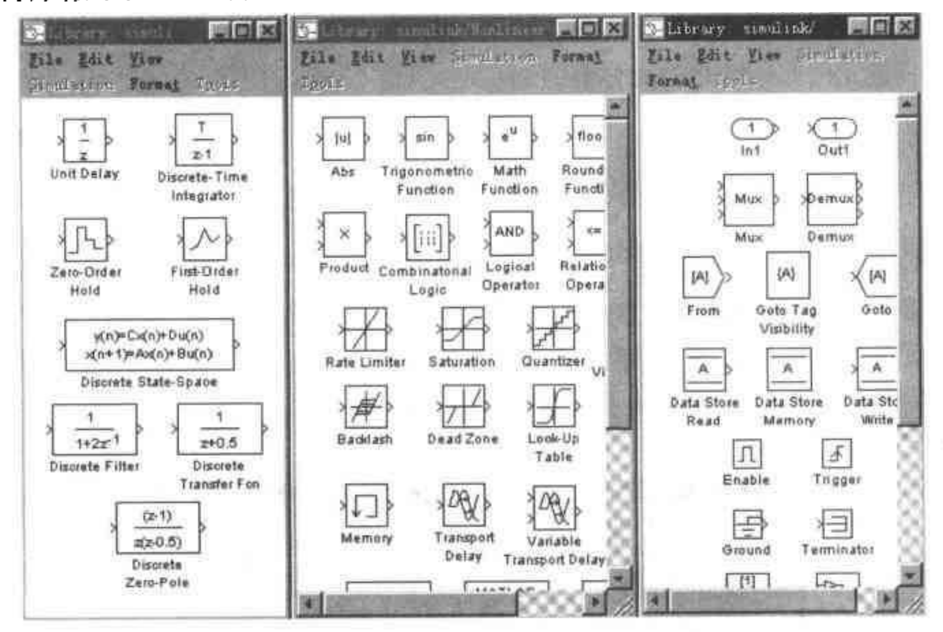


图 6－13　离散系统、非线性系统和连接装置元件库

离散系统元件库中常用的元件有:

(1) 标有"Unit Delay"的 ,单位延迟元件。这是构成离散控制系统的最基本元件,其作用是将该单元的输入量延迟一个单位时间再输出。

(2) 标有"Discrete Transfer Fcn"的 ,离散系统传递函数元件。双击该元件也可以设置传递函数的系数,与 MATLAB 环境下相同,其格式也是传递函数分子和分母多项式系数的降幂排列。

(3) 标有"Discrete State－Space"的 ,离散系统状态空间描述元件。双击该元件也可以设置状态空间描述系数矩阵(C,D,A,B)。

非线性系统元件库中常用的元件有:

(1) 标有"Product"的 ,乘法器元件。该元件可以将两路或多路输入信号相乘,

设计非线性控制器时非常方便。

(2) 标有"Saturation"的 ，饱和非线性元件。这是非线性系统中使用较多的一个元件。事实上，一般的放大器和控制器阀门的特性都可以用该元件描述。双击该元件可以设置其斜率和上下限值。

(3) 标有"Dead Zone"的 ，死区非线性元件。一般有铰链和转轴的系统的相关部位可以用该元件来近似，电动机的精确模型中也要用到该元件。双击该元件可以设置其死区宽度和斜率。

(4) 标有"MATLAB Fcn"的 ，MATLAB 函数元件。该元件可以将输入值传递到 MATLAB 函数中处理，并将返回值输出。双击该元件可以设置相应的 MATLAB 函数名称。

连接装置元件库中常用的元件有：

(1) 标有"Mux"的 ，多路开关元件。该元件可以将多路的输入向量以某种方式合并后从一路输出，具体的合并方式可由双击该元件来设置。

(2) 标有"Demux"的 ，多路解耦元件。该元件可以将单路的输入向量以某种方式解耦后多路输出，具体的解耦方式可由双击该元件来设置。

这些元件库中还有许多非常有用的元件，这里就不一一列举了。如果读者希望深入了解，可以选中某个元件后双击帮助选项，即可进入 SIMULINK 元件帮助界面，如图 6－14所示。

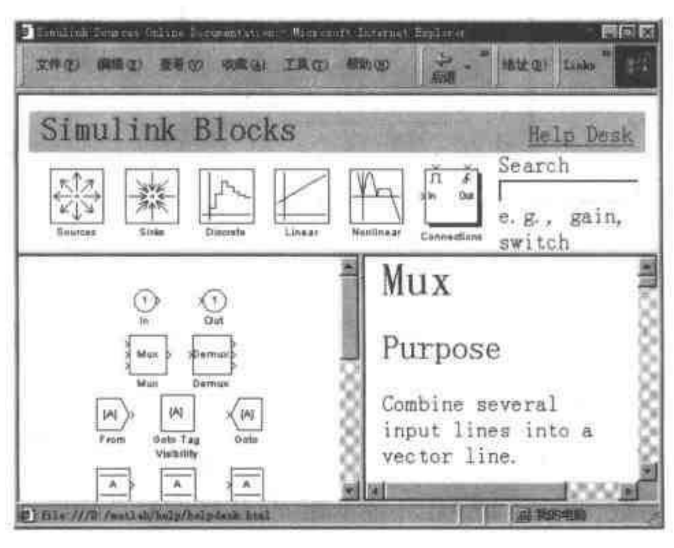


图 6－14 SIMULINK 元件库帮助界面

该帮助界面是以 Web 浏览器的方式给出的，用鼠标点击相应的元件库或元件就可以浏览其帮助内容。

在图 6－11 所示的 SIMULINK 元件库的左下角还有一个"Blocksets & Toolboxs"图标，双击该图标，即可进入 SIMULINK 支持的模块库和工具箱：

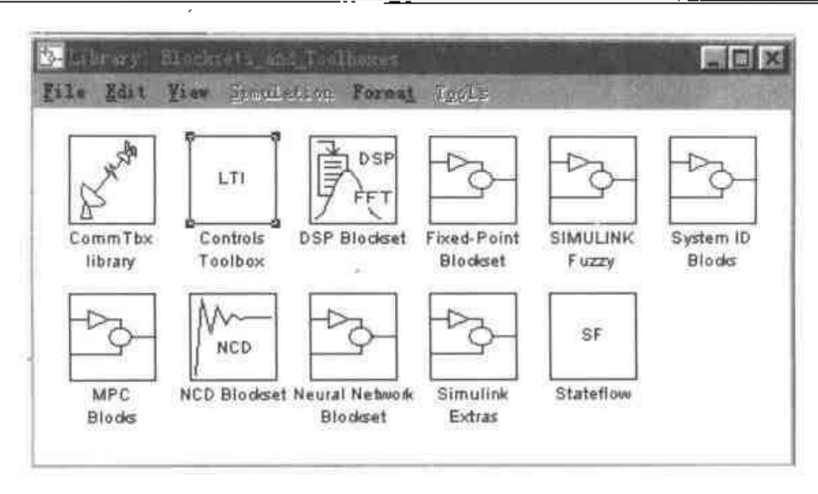


图 6-15　SIMULINK 支持的模块库和工具箱

该元件库中包括通信工具箱元件库(commTbx library)、控制系统工具箱(Controls Toolbox)、离散信号处理模块库(DSP Blocxset)、定点模块库(Fixed-Point Blockset)、模糊仿真(SIMULINK Fuzzy)、系统辨识模块(System ID Blocks)等等。其中我们经常用到的是标有"Controls Toolbox"的 ，控制系统工具箱模块。双击该图标，即可进入 LTI 系统模块界面：

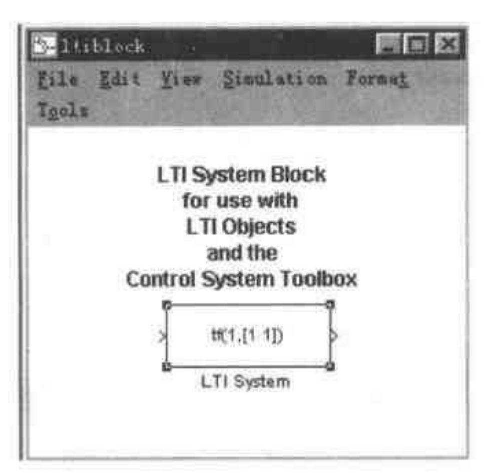


图 6-16　LTI 系统模块界面

将该界面下的 LTI System 模块拖到(或拷贝到)SIMULINK 仿真环境下，就可以设置线性定常系统。包括离散系统和连续系统，传递函散描述和零极点描述、状态空间描述等等各种线性定常系统的模型建立和仿真都可以通过该模块来进行。双击该图标，即进入 LTI 系统设置界面：

所有控制系统工具箱支持的控制系统描述方式都可以在该界面中设置，只要给出相应的输入值即可。例如图 6-17 中"LTI system variable"文本框中输入的是系统的传递函数描述模型"tf(1;[1,1])"。

在图 6-11 所示的 SIMULINK 元件库的右下角还有一个"Demo"图标，双击该图标，即可进入 SIMULINK 演示程序界面。

事实上，这是一个非常好的学习 SIMULINK 的演示软件。通过该软件，读者可以了解 SIMULINK 的各项主要功能和用法。其中包括：

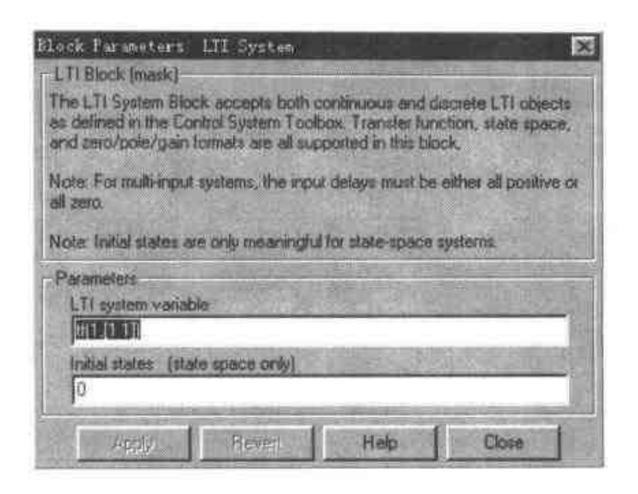


图 6-17 LTI 系统设置界面

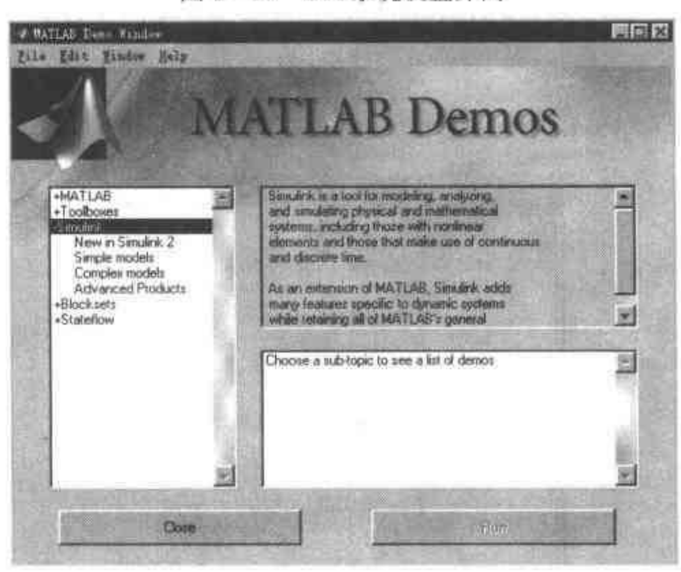


图 6-18 SIMULINK 演示程序界面

(1) New in Simulink 2,介绍 SIMULINK 2 的最新的特性和功能。和 MATLAB 4.x 内置的 SIMULINK 1 相比,基于 MATLAB 5.2 的 SIMULINK 2.2 有了许多新的特性。因此,虽然原来的函数和模块基本上都可以使用,但仅仅熟悉 SIMULINK 1 的读者最好还是阅读一下该选项的内容。

(2) Simple models & Complex models,这是基于简单模型和复杂模型介绍 SIMULINK 的例子。不过虽然有 Simple models 的选项,这里面的例子对于初学者来说,还是过于复杂。本书将根据前几章里一些典型的例题类型重新编写一些适合初学者的例子,争取使读者看懂了这些例子后就能阅读 SIMULINK 下自带的 Simple models& Complex models。

(3) Advanced Products,这里面介绍的是"Real-Time Workshop",其功能是根据 SIMULINK 搭建的系统方框图自动生成 C 代码,并提供给各种仿真界面使用。对于大型控制系统来说,由于 C 语言是编译执行的(MATLAB 相当于解释执行的高级语言),生成 C 的源代码可以保证仿真运行的实时性。不过对于小的控制系统来说,MATLAB 环境已经足以保证仿真实时性了。

3. SIMULINK 仿真环境参数设置

在图 6－3 的 SIMULINK 仿真环境下，我们可以调整各种的仿真参数，包括仿真算法和仿真精度，从而获得比较满意的仿真效果。选择该界面下的“Simulation”菜单的“Parameters”选项：

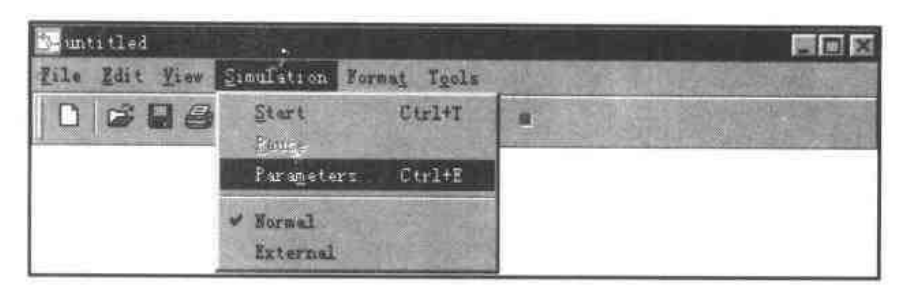


图 6－19　进入 SIMULINK 参数设置界面

单击该菜单选项或按下“Ctrl－E”即可进入 SIMULINK 仿真环境参数设置界面：

图 6－20　SIMULINK 仿真环境参数设置界面

该界面下有五个选择对话框，缺省的也是最常用的是图 6－20 所示的“Solver”求解对话框。

图 6－20 中的下拉菜单中的选项是仿真算法，缺省的是“ode45”，Dormand－Prince 算法。其余的如 ode23、ode113、ode15s 等我们在第二章讲述求解微分方程的 MATLAB 函数时都曾经分绍过。

图 6－20 第一行“Simulation Time”选项中设置的是仿真的开始和结束的时间，缺省是0 和 10。如果读者使用的是 SIMULINK 1 的话尤其要注意这个选项，因为 SIMULINK 1 中设置的缺省结束时间是 9999，如果不改成希望的时间，系统就会不间断的仿真下去，并且求得的曲线也因此比例失调。

图 6－20 的下拉菜单挡住了后面设置仿真精度的选项。缺省的相对精度（Relative tolerance）是 1e－3，绝对精度（Absolute tolerance）是 1e－6，读者可以根据不同的仿真精度要求按照科学记数法的形式自行设置。

此外,在图 6-3 的 SIMULINK 仿真环境下选择"Format"菜单,还可以进行设置仿真背景和前景的颜色、线条的宽度和颜色、修改元件名称等各种样式的设置,如图 6-21 所示。

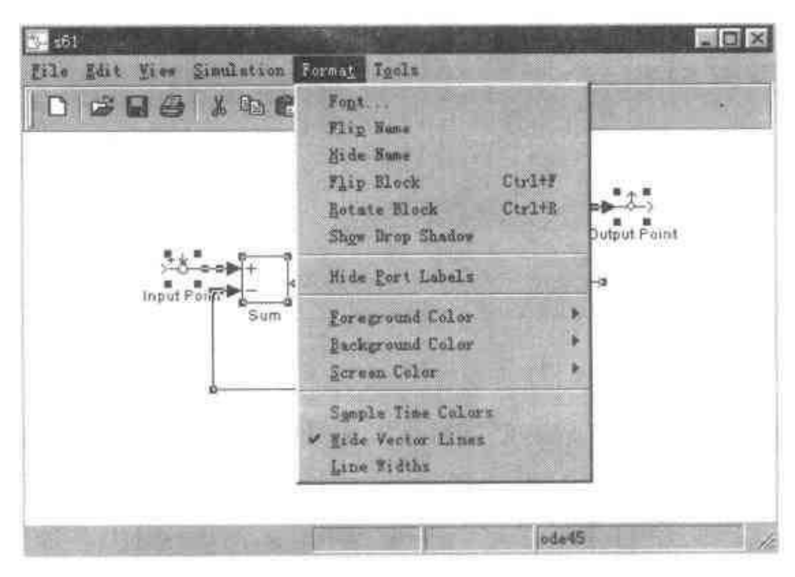


图 6-21 SIMULINK 环境下仿真样式的设置

例如,我们可以选择"Flip Name"选项将元件的名称放到元件的上方;或者选择"Show Drop Shadow"选项为选中的元件加阴影。这些选项的功能都比较琐碎,实现也比较简单,因此这里就不介绍了,在实际应用过程中读者很快就会掌握这些功能。

关于 SIMULINK 的基本功能就介绍到这里,其他一些例如如何建立新文件、如何存盘、如何打印图形的常识性功能这里就不一一介绍了。至于 SIMULINK 环境下许多深入的功能,我们留到下一章再介绍,还有一些,就需要读者自己摸索,利用本节介绍的帮助手段,逐步提高了。

6.2 SIMULINK 下控制系统分析

在前五章中,我们曾经系统的介绍了 MATLAB 文字界面下控制系统的分析和设计问题。转到可视化的 SIMULINK 界面下,这些问题的的解题思路没有什么变化,但是解题手段更加丰富、更加直观了。

一般说来,SIMULINK 环境适于构建大型控制系统,尤其是包含非线性环节的控制系统。因为它可以提供各种集成化的模块,并且结构清晰明了。也正因为如此,SIMULINK 演示程序中的例子都比较复杂,对于初学者来说不太适合。上一节我们介绍了有关 SIMULINK 环境的基础知识,本节准备应用这些基础知识,通过几个较为详细的例子,进一步加深读者对 SIMULINK 可视化解决问题思路的理解。

6.2.1 传递函数模型

上一节例 6.1 也是一个有关传递函数模型仿真的例子。不过那个例子比较简单,主

要是为了让没有接触过 SIMULINK 的读者对它有一个感性的认识。本节准备介绍一个过程控制系统中有关 PID 校正装置设计的例子,详细讨论一下利用 SIMULINK 构筑系统传递函数模型的优点。

[例 6.2] 过程控制系统中常用的 PID 校正装置传递函数为:

$$G(s) = K_P + \frac{K_I}{s} + K_D s$$

其中 Kp、Ki、Kd 分别是比例系数、积分系数和微分系数。第四章我们曾经详细介绍过 MATLAB 环境下 PID 校正装置参数整定的方法和实例。在 MATLAB Command Window 下,可以通过编程绘图不断校正 PID 参数,最终求得满意的结果。而在 SIMULINK 环境下,借助其方框图搭建非常方便、仿真参数可以随时修改的优点,我们还可以采用下述的稳定边界法。

表 6-1 稳定边界法参数整定计算公式

调节规律 \ 整定参数	Kp	Ki	Kd
P	0.5Kp		
PI	0.455Kp	0.535Kp/T	
PID	0.6Kp	1.2Kp/T	0.075KpT

使用稳定边界法整定 PID 参数分为以下几步:

1. 将积分系数 Ki 和微分系数 Kd 设为 0,Kp 置较小的值,使系统投入稳定运行。若系统无法稳定运行,则选择其他校正方式。

2. 逐渐增大比例系数 Kp,只到系统出现等幅振荡,即所谓临界振荡过程。记录此时的临界振荡增益 Kp 和临界振荡周期 T。

3. 按照表 6-1 所示的经验公式和校正装置类型整定相应的 PID 参数,然后再进行仿真校验。

在 SIMULINK 环境下采用稳定边界法的优点是非常直观,完全可视化操作,省去了编程的工作量。当然,采用稳定边界法整定的 PID 参数未必是最优的,读者也可以借助 SIMULINK 仿真环境自己设计一套 PID 参数整定方法。

试采用稳定边界法整定控制系统 PID 调节器。已知某被控对象为二阶惯性环节,其传递函数为:

$$G(s) = \frac{1}{(5s+1)(2s+1)}$$

测量装置和调节阀特性为:

$$G_m(s) = \frac{1}{10s+1}, G_v(s) = 1.0。$$

结果:最终整定的 PID 校正装置参数为:

```
Kp = 7.5000
Ki = 0.5000
Kd = 14.2500
```

分析:本例是一个过程控制对象,其特点是时间常数较大、控制要求不是很精确。因

此,PID 校正是最主要的手段。而在 SIMULINK 环境下,应用稳定边界法整定 PID 参数是非常方便的。

求解过程:

本例的求解分为以下几步:

1. 搭建系统方框图

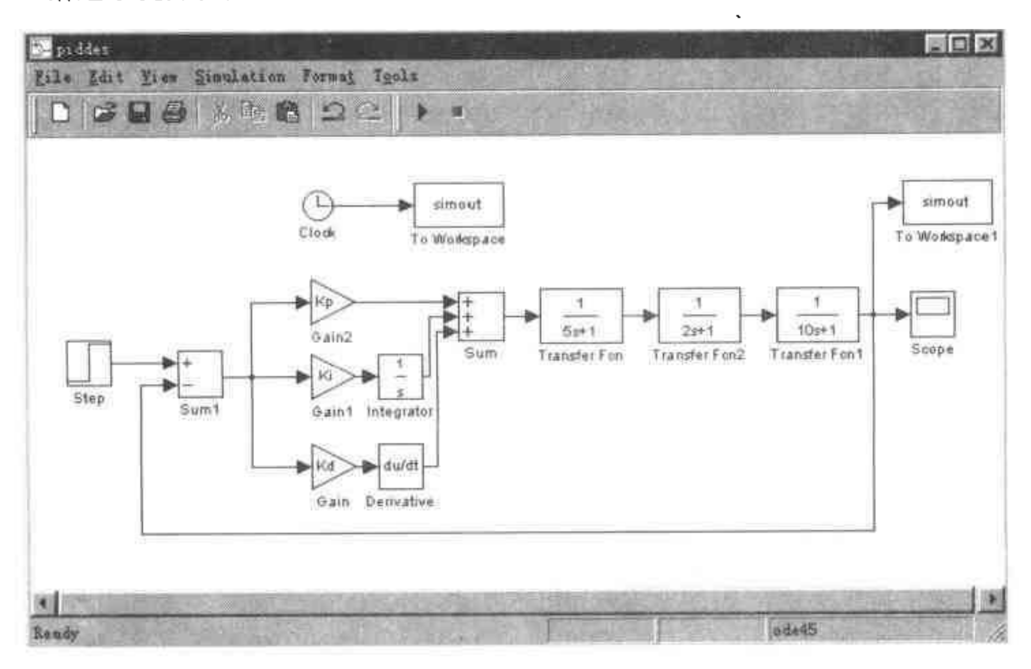


图 6-22 PID 校正系统方框图

进入 SIMULINK 仿真环境后搭建如图 6-22 所示的 PID 校正系统方框图。该方框图包含的元件有:线性系统元件库的"Transfer Fcn"元件三个,"Sum"元件两个,"Gain"元件三个,"Integrator"、"Derivative"元件各一个;输入源元件库的"Step"元件一个,"Clock"元件一个;输出方式元件库的"Scope"元件一个,"To Workspace"元件一个。

2. 设置 PID 参数名称及环境参数

分别双击 PID 校正装置的三个"Gain"元件,在弹出的对话框里填入相应的 Kp、Ki、Kd:



图 6-23 PID 参数名称设置

图 6-23 是与 Kp 相关的"Gain"元件参数设置对话框,其余 Ki、Kd 参数设置对话框均与此类似。

打开图 6－22 仿真环境的“Simulation”菜单，选择“Parameters”选项，将“Stop Time”设置为 60，“Relative tolerance”设置为 1e－5：

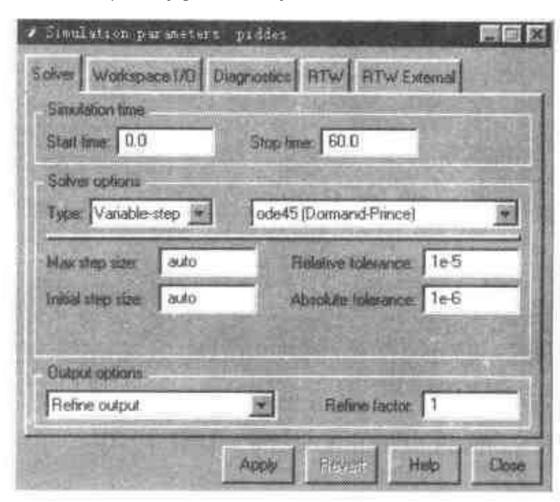


图 6－24 SIMULINK 环境参数设置

一般来说，仿真结束时间和仿真相对增益是根据不同的控制对象来选择的。因为本例是过程控制对象，所以仿真时间选的长一些。而将仿真精度定的高一些，是为了让曲线画的更清楚。

分别双击两个“To workspace”元件的图标，将与“clock”元件相连的“To workspace”元件的输出变量设置为 tout；与系统输出相连的“To workspace”元件的输出变量设置为 simout。这是输出到 MATLAB Workspace 下的变量名称。第一个“To workspace”元件参数设置界面见图 6－25，第二个元件参数设置与此类似。

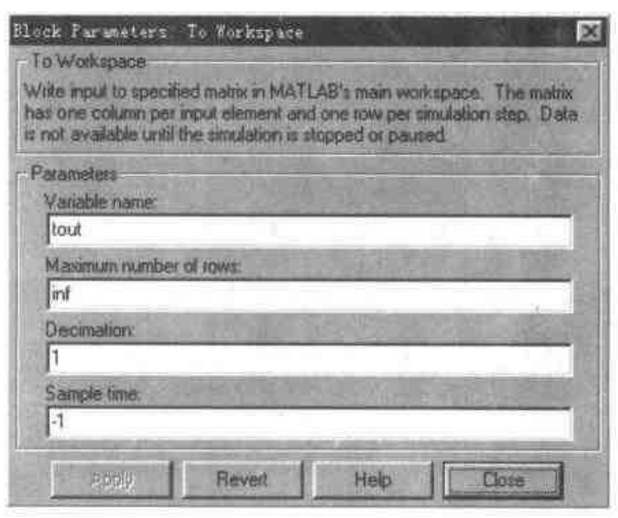


图 6－25 To Workspace 元件参数设置

该元件的作用是将仿真时间和仿真结果输出到 MATLAB 环境下的 Workspace，然后我们就可以应用相应的变量以及各种 MATLAB 函数进行分析了。

3．在 MATLAB 环境下初始化相应的 PID 参数变量

如果刚搭建好系统的方框图就进行仿真的话，读者就会看到以下的对话框：

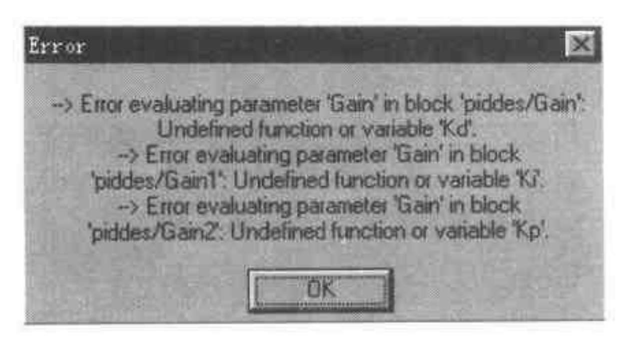


图 6-26 SIMULINK 错误对话框

系统提示说变量 Kp、Ki 和 Kd 没有定义，无法进行仿真。这是因为 SIMULINK 是根据数值求解的，因此使用变量，必须在相应的 Workspace 中有定义。我们仅仅给出了变量名而没有赋值，系统自然无法仿真。

我们回到 MATLAB Command Window，键入以下代码：

```
Kp = 1;
Ki = 0;
Kd = 0;
```

回到 SIMULINK 环境下就可以开始仿真了。当然，也可以在"Gain"元件参数设置对话框中直接键入相应的数值，但这样一来物理意义就不够明显，并且不利于以后的结构化建模和模块设置。

4. 整定 PID 参数。

下面我们按照稳定边界法整定 PID 参数。稳定边界法的第一步参数在进行 PID 参数变量初始化已经设置好了，因此，点击图 6-22 仿真环境的 按钮或选择"Simulation"菜单的"Sart"选项，得到仿真曲线如图 6-27 所示。

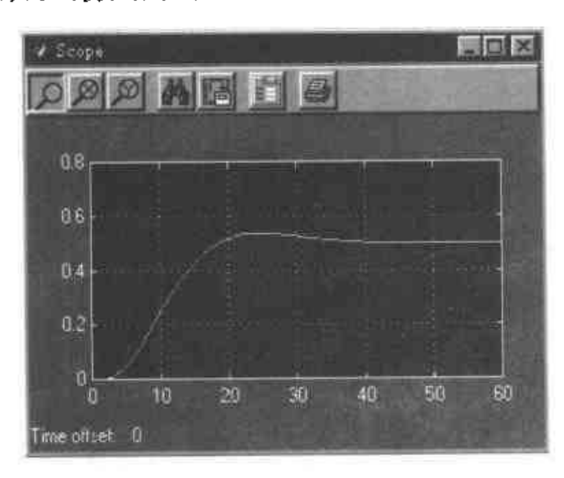


图 6-27 系统阶跃响应曲线

从图中我们可以看到，系统能够稳定运行，因此可以用稳定边界法来整定 PID 参数。希望通过 PID 校正，能够使系统静态无差，并且改善其快速性。在进行稳定边界法的第二步之前，我们先看一下当前的 Workspace 中有什么变量。在 MATLAB Command Window 下键入"Whos"：

```
whos
Name        Size        Bytes       Class
Kd          1x1         8           double array
Ki          1x1         8           double array
Kp          1x1         8           double array
simout      162x1       1296        double array
tout        162x1       1296        double array
Grand total is 491 elements using 3928 bytes
```

此时我们看到，PID 参数 Kp、Ki 和 Kd，系统仿真时间 tout 和仿真结果 simout 都在当前的 Workspace 下，可以用各种 MATLAB 函数进行仿真和分析。

稳定边界法的第二步是求取临界振荡周期和临界增益。方法是逐步增大 Kp，只到系统出现等幅振荡为止。记下此时的 Kp，即为临界增益；此时的曲线两峰值之间的距离即为临界振荡周期。

具体在 SIMULINK 下可以如下实现：先选取较大的 Kp，例如 100（不同的对象有不同的值），使系统出现不稳定的增幅振荡；然后采取折半取中的办法寻找临界增益，例如第一个点是 Kp = 50，如果仍增幅振荡则选下一个点 Kp = 25，否则选取 Kp = 75。如此不断对折可以最快找到临界增益。当然，经验丰富的工程人员可以根据曲线的振荡情况很快的找到临界增益。

利用 MATLAB 求解非线性方程的功能也可以精确的确定临界增益和临界振荡周期。不过一来非常准确的精确值没有必要（稳定边界法本身就是经验公式而非解析解），二来重新编写求解非线性方程的 M 函数也要耗费一定的时间，还不能保证方程列写的准确性。因此，我们直接在 SIMULINK 下通过对折取中的方法求出近似值。具体的寻找过程这里就略去了，等幅振荡时，有：

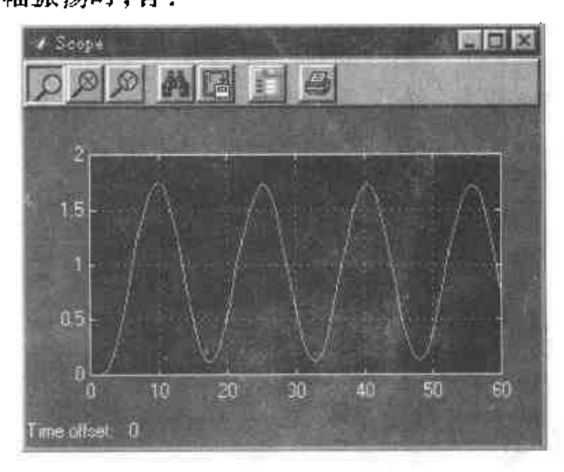


图 6 – 28 等幅振荡的阶跃响应曲线

此时 Kp = 12.5，从图 6 – 28 中读出曲线两峰值之间的距离约为 15s 多一点，因此取 T = 15.2s。下一步即可算出按稳定边界法表 6 – 1 整定的 PID 参数了。在 MATLAB Command Window 下键入以下代码：

```
%临界增益
Kp = 12.5;
```

```
%临界振荡周期
T = 15.2;
%PID参数设置
Kd = 0.075 * Kp * T;
Ki = 1.2 * Kp/T;
Kp = 0.6 * Kp;
```

其中第一个 Kp=12.5 是用折半取中法寻找出来的，这里为了保持代码的完整性也把它列出来了。此时的 PID 参数为

```
Kp = 7.5000
Ki = 0.9868
Kd = 14.2500
```

系统的阶跃响应曲线如图 6-29。

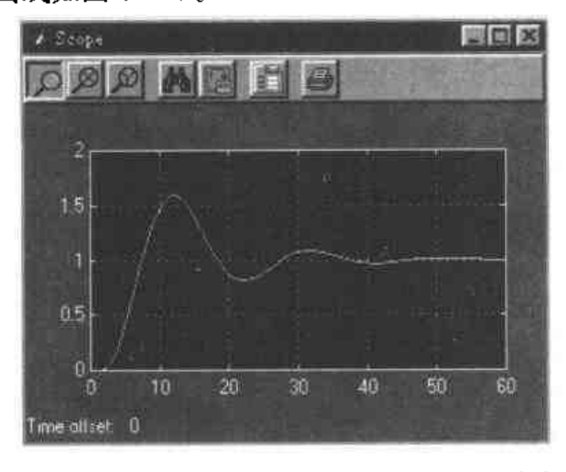


图 6-29 稳定边界法整定的系统阶跃响应曲线

有的读者可能会问，该系统的响应曲线过渡过程时间超过 30s，超调量大于 50%，和以前的系统相比，控制品质并不令人满意。这是因为，以前讲述的系统都是机械或电子控制系统，而本例的一般过程控制系统对过渡过程时间和超调量要求并不是非常严格。当然，超过 50% 的超调量还是大了一些，为此我们降低积分系数 Ki，重新进行仿真，有：

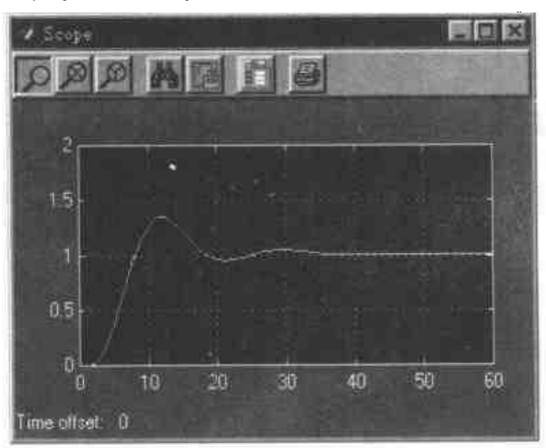


图 6-30 校正后系统的阶跃响应曲线

此时，Ki=0.7，Kp 和 Kd 仍是由稳定边界法整定的数据。从图 6-30 中可以看出，

系统的过渡过程时间和超调量都有所降低,对于没有特殊要求的过程控制系统来说,这样的性能指标已经可以满意了。

5. MATLAB 环境下分析

前文我们曾经提到过,通过"To Workspace"元件可以将 SIMULINK 下的仿真时间和仿真结果传到 MATLAB 的 Workspace 中去。因此,建立好模型后我们也可以在 MATLAB Command Window 下进行分析。例如,运行完程序后键入"plot(tout,simout)",就可以绘制出此时的仿真曲线如图 6-31。

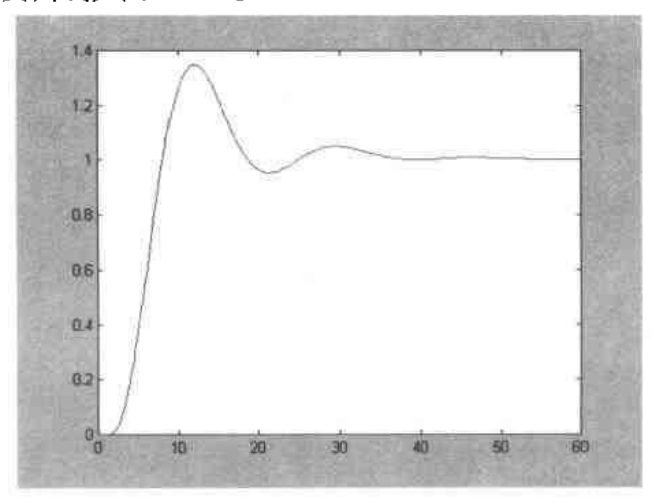


图 6-31 系统的阶跃响应曲线

可以看出,结果和在 SIMULINK 下绘制的曲线一样,不过 MATLAB 下绘制的曲线更加美观和清晰一些,而且可以直接拷贝到其他的 Windows 应用程序中。并且,还有许多 MATLAB 函数可以使用。

小结:本例比较详细的分析了 SIMULINK 环境下控制系统传递函数模型系统的搭建和仿真问题。读者通过这个例子可以更进一步领略 SIMULINK 可视化设计的思路和强大的功能。事实上,SIMULINK 还自动生成方框图的 MATLAB 代码。例如,我们在 MATLAB Editor/Debugger 下打开本例"piddes.mld",可以看到:

```
Model {
Name                "piddes"
Version             2.20
SimParamPage        Solver
SampleTimeColors    off
InvariantConstants  off
WideVectorLines     off
ShowLineWidths      off
StartTime           "0.0"
StopTime            "60.0"
Solver              ode45
......
BlockDefaults {
Orientation         right
```

```
        ForegroundColor         black
        BackgroundColor         white
        DropShadow              off
        NamePlacement           normal
        FontName                "Helvetica"
        FontSize                10
        FontWeight              normal
        FontAngle               normal
        ShowName                on
        }
        ......
    System {
        Name                    "piddes"
        Location                [86, 139, 744, 424]
        Open                    on
        ToolBar                 on
        StatusBar               on
        ScreenColor             white
        PaperOrientation        landscape
        PaperPositionMode       auto
        PaperType               usletter
        PaperUnits              inches
        Block {
          BlockType                   Clock
          Name                        "Clock"
          Position                    [185, 35, 205, 55]
          Location                    [283, 500, 390, 545]
        }
        ......
        Block {
          BlockType                   TransferFcn
          Name                        "Transfer Fcn"
          Position                    [345, 102, 400, 138]
          Numerator                   "[1]"
          Denominator                 "[5 1]"
        }
        ......
        Line {
          SrcBlock                    "Transfer Fcn2"
          SrcPort                     1
          DstBlock                    "Transfer Fcn1"
          DstPort                     1
```

……

因为空间有限,只列出了重要的代码段。代码段开始是有关模型的一些初始化信息,例如模型名称、版本号、仿真起止时间以及仿真算法等等。然后是元件块的初始化信息,例如元件输出端方向、前后景颜色、元件的线型以及字体等等。紧接着是程序的主体:系统描述代码段,包括系统初始化信息、元件类型及内容以及各元件的连接信息等等。例如在第二个 Block 代码段里可以看到我们熟悉的“Numerator "[1]" Denominator "[5 1]"”它描述的是图 6-22 中的传递函数元件 $G(s)=\frac{1}{5s+1}$。

虽然一般我们不会自己用手工编写 SIMULINK 模型代码,但了解了 SIMULINK 模型的结构和编写方式对于我们更加深入的学习 SIMULINK 环境,使用 SIMULINK 模型仿真都是很有用处的。

6.2.2 状态空间模型

一般说来,SIMULINK 比较适合传递函数模型描述的控制系统仿真和设计。不过对于状态空间模型,SIMULINK 同样可以对其进行分析和仿真。只是如果进行状态变量的分析和运算,需要重新布局。请看下例:

系统的状态方程和输出方程如下。试在 SIMULINK 环境下对系统的开环阶跃响应进行仿真,并设计状态反馈向量和参考输入,将系统的闭环极点配置在 -2 和 -4。

$$A=\begin{bmatrix}-2 & 1\\ 1 & -2\end{bmatrix} \qquad B=\begin{bmatrix}1\\ 0\end{bmatrix}$$

$$C=[1 \quad -1] \qquad D=0$$

结果:系统的开环阶跃响应见图 6-32。

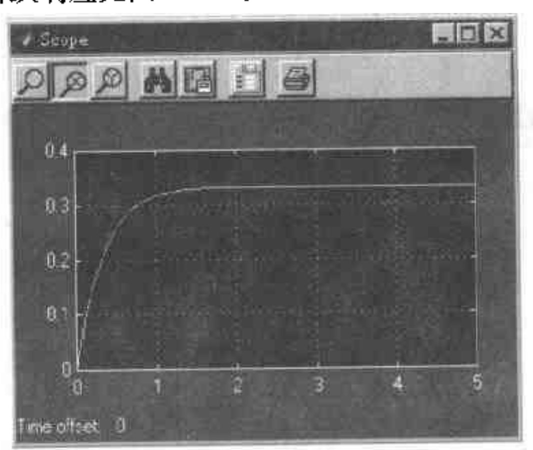


图 6-32 系统的开环阶跃响应曲线

求得的状态反馈矩阵和参考输入为:

```
K = 2     1
Nbar = 8.0000
```

分析:SIMULINK 线性系统元件库提供了仿真控制系统状态空间描述的元件“Stete-Space”,可以利用该元件对此二阶系统进行仿真。不过该元件的输出端是系统输出量而非状态变量,因此,设计状态反馈时我们要对系统的结构作一些调整。

[例 6.3] 求解过程:本例的求解分为以下几步

1. 求解系统开环阶跃响应

在 SIMULINK 仿真环境下搭建如图 6-33 所示的方框图,包括一个阶跃函数"step"元件,一个状态空间描述"Stete-Space"元件,一个输出的"scope"元件。其中"Stete-Space"元件可以在线性系统元件库中找到。

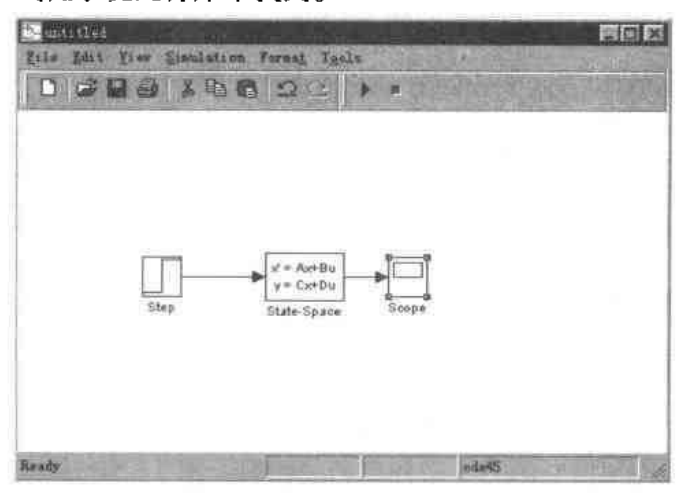


图 6-33 二阶系统方框图

选择"Simulation"菜单的"Parameter"选项,将仿真结束时间设置为 5s,然后选择"Start"选项,得到系统的开环阶跃响应曲线如图 6-32 所示。

2. 搭建系统的状态反馈模型

从图 6-32 也可以看出,SIMULINK 环境下状态空间模型的输出端仅有系统的输出量 y。因此我们改变一下系统的结构,使得两个状态变量 X 都能作为反馈信号。搭建新的系统方框图如图 6-34。

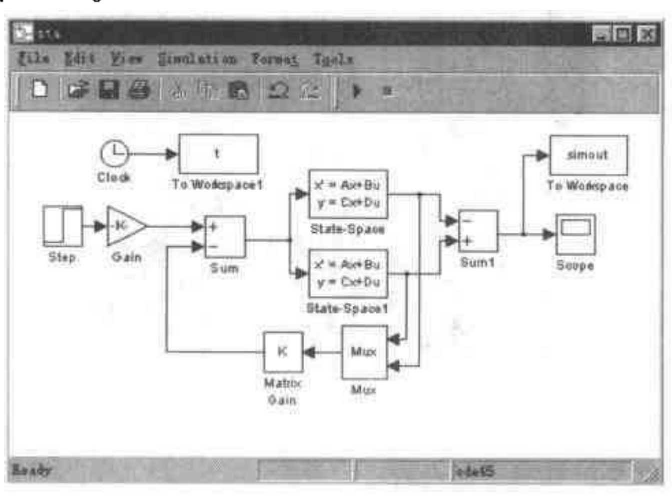


图 6-34 含有状态反馈的系统方框图

其中"Mux"元件属于连接模块元件库,功能是将多路输入合成一路输出。"Matrix Gain"元件属于线性系统元件库,功能是将输入的矩阵与增益矩阵相乘后输出。为了使该系统有一定的通用性,可以在 MATLAB Command Window 下设置系统的各参数:

```
clear
A=[-2 1;1 -2];
B=[1;0];
C=[1 -1];
P=[-2 -4];
K=place(A,B,P);
Nbar=rscale(A,B,C,0,K);
```

其结果为：

```
place: ndigits = 15
K = 2     1
Nbar = 8.0000
```

图 6-34 中两个“Stete-Space”元件的输出量分别是系统的两个状态变量。它们通过一个“Sum”元件拟合 C=[1 -1]向量(如果元素值不为 1 可以加入“Gain”元件)。双击该“Stete-Space”元件，可以设置系统状态空间模型：

由于我们已经在 MATLAB 当前的 Workspace 中设置了系统的状态空间模型的变量 A,B,C,所以在图 6-35 的“Parameter”选项中可以直接填写变量名。否则的话就应该按照 MATLAB 环境下输入矩阵的格式填写各参数矩阵。



图 6-35　设置系统的状态空间模型

C 矩阵中填写“[1 0]”是为了取得第一个状态变量，使输出量中第二个分量为零。同理，在另一个“Stete-Space”元件的相应位置填写完 A、B 后，在 C 矩阵中填写“(0 1)”，取得第二个状态变量。

双击“Matrix Gain”元件，在其“Parameter”复选框的“Gain Matrix”对话框中填入上一步设计好的状态反馈矩阵 K，如图 6-36 所示。

注意此时的参数是矩阵“K”，而非标量。

双击“Gain”元件，在其“Parameter”复选框的“Gain”对话框中填入上一步设计好的参考输入倍数 Nbar，如图 6-37 所示。

关于状态反馈矩阵和参考输入的具体含义请参看第五章的相关部分。

至此，有关状态反馈系统的方框图搭建和参数设置就基本完成了。感兴趣的读者还



图 6－36　设置矩阵增益元件参数

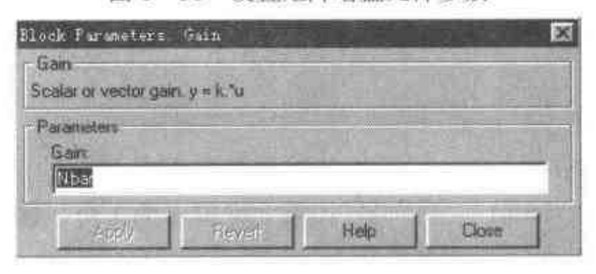


图 6－37　设置增益元件参数

可以设置一些类似仿真精度、曲线线型、前后景颜色等参数，使绘制的曲线更加美观。这里就不一一列举了。

在 SIMULINK 仿真环境下点击 按钮或选择“Simulation”菜单的“Start”选项，得到仿真曲线如图 6－38 所示。

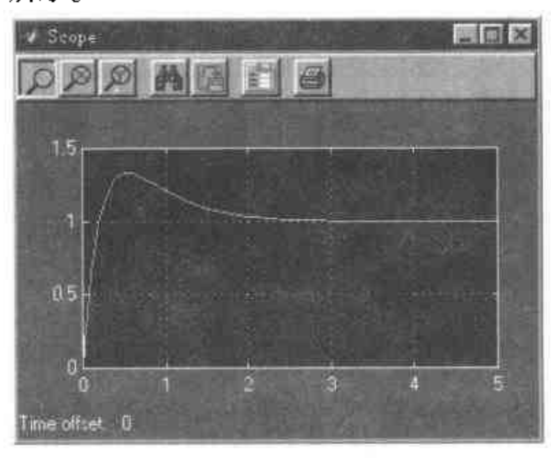


图 6－38　状态反馈系统阶跃响应曲线

相应的，在 MATLAB 环境下我们也可以得到系统的仿真参数：

whos

Name	Size	Bytes	Class
A	2x2	32	double array
B	2x1	16	double array
C	1x2	16	double array
K	1x2	16	double array
Nbar	1x1	8	double array
P	1x2	16	double array
simout	64x1	512	double array

```
tout               64x1         512          double array
Grand total is 205 elements using 1640 bytes
```

以及系统的闭环阶跃响应曲线如图 6－39 所示。

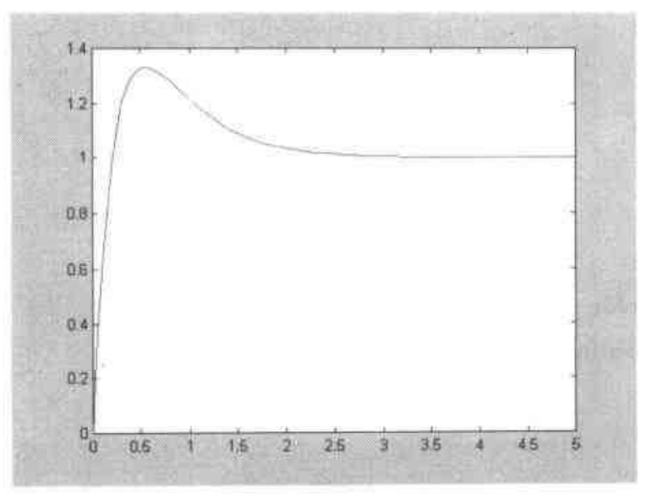


图 6－39　plot()函数绘制的系统阶跃响应曲线

可以看到，通过闭环极点配置，系统的闭环响应性能指标有了一定改善。并且，只需在 MATLAB 环境下修改 P 向量就可以重新配置系统的闭环极点，获得特定要求的性能指标。本例的方框图有一定代表性，可以用这种形式来获得系统的状态向量，设计状态反馈和参考输入。

小结：本例分析了 SIMULINK 环境下控制系统状态空间模型系统的搭建和仿真问题，给出了状态空间反馈的一种初步解法。事实上，SIMULINK 环境下还有其他更好的搭建状态反馈模型的方法，限于篇幅这里就不再介绍了。对于初学者来说，SIMULINK 环境更适用于系统的传递函数模型，当然，应用状态空间模型同样可以获得满意的效果。

同样，我们在 MATLAB Editor/Debugger 下打开本例“sta. mld”，可以看到该状态反馈模型的代码：

```
Model {
  Name                  "sta"
  Version               2.20
  SimParamPage          Solver
  SampleTimeColors      off
  InvariantConstants    off
  WideVectorLines       off
  ShowLineWidths        off
  StartTime             "0.0"
  StopTime              "5"
......
AnnotationDefaults {
  HorizontalAlignment   center
  VerticalAlignment     middle
```

```
    ForegroundColor         black
    BackgroundColor         white
......
System {
    Name                    "sta"
    Location                [70, 200, 570, 460]
    Open                    on
    ToolBar                 on
    StatusBar               on
    ScreenColor             white
    PaperOrientation        landscape
......
Block {
    BlockType               Reference
    Name                    "Matrix\nGain"
    Ports                   [1, 1, 0, 0, 0]
    Position                [195, 165, 225, 195]
    Orientation             left
    SourceBlock             "simulink/Linear/Matrix\nGain"
    SourceType              "Matrix Gain"
    K                       "K"
}
Block {
    BlockType               Mux
    Name                    "Mux"
    Ports                   [2, 1, 0, 0, 0]
    Position                [255, 160, 290, 200]
    Orientation             left
    Inputs                  "2"
}
......
Block {
    BlockType               StateSpace
    Name                    "State-Space1"
    Position                [230, 102, 290, 138]
    A                       "A"
    B                       "B"
    C                       "[1 0]"
    D                       "0"
```

```
    X0                     "0"
  ......
```

上面代码中有关系统设置和仿真参数的部分在例 6.2 中已经介绍过，这里就不再重复了。最后一段代码是描述“State - Space”元件的，包括元件类型、名称、位置、参数设置等等。

此外，有的读者在搭建图 6 - 34 的状态反馈方框图时有可能会遇到“Martix Gain”元件和“Mux”元件输入端和输出端位置相反的问题，无法用正常的直线连接形成反馈。选中该元件，单击鼠标右键，选择“Format” - “Rotate”即可解决此问题。相应的修改菜单见图 6 - 41，图 6 - 40 是翻转的结果。

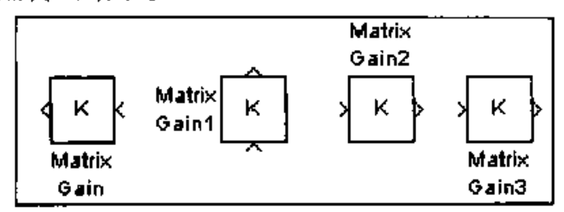


图 6 - 40 翻转元件及改变其名称位置

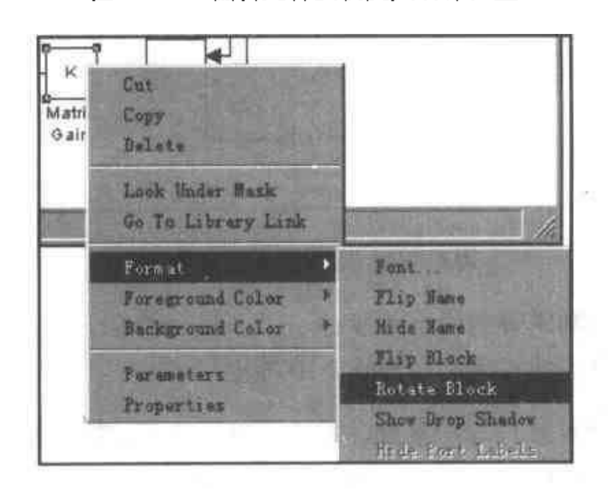


图 6 - 41 改变元件的方向

每次点击“Rotate”选项元件翻转 90°。还可以选择“Flip Name”选项改变元件名称的位置，具体的结果见图 6 - 40。

6.2.3 非线性模型

非线性对象是指其运动规律要用非线性代数方程和(或)非线性微分方程描述，而不能由线性方程组描述。控制系统中若含有非线性部件，则称为非线性控制系统。

本书在第一章里曾经提到，几乎所有的实际的控制系统中，都有非线性部件，或是部件特性中含有非线性。在一些系统中，人们甚至还有目的的应用非线性部件来改善系统性能和简化系统结构。因此，严搭地讲，几乎所有的系统都是非线性的。

SIMULINK 环境下有非线性系统元件库，因此，进行非线性系统仿真非常方便。如

果忽略具体的仿真细节，在 SIMULINK 环境下进行非线性系统仿真和线性系统仿真几乎没有什么区别，都是根据系统结构选择相应的元件搭建系统的方框图，然后选择参数仿真，根据仿真结果进行设计或校正。

[例 6.4] 试选取相应的非线性元件设计实际的 PID 控制器，满足下述要求：

1. 当 PID 控制器输出的绝对值超过某一给定的上限时，积分器停止作用。

2. 微分器对阶跃输入的响应应持续一段时间。

将设计好的控制器与理想 PID 控制器比较。

分析：在广泛应用 PID 控制器的过程控制领域，实际应用的一般都是本例介绍的实际 PID 控制器。一方面是考虑到系统的物理可实现性，必须用实际微分器代替理想的微分装置；另一方面实际的控制器阀门都有上下限，不可能无限积分。因此，必须设计出实用的 PID 控制器。积分上下限可以用饱和非线性元件实现，SIMULINK 环境下也提供了该元件的模型。实际微分器可以用分子大于分母的一阶传递函数实现。

结果：设计好的实际 PID 控制器如图 6－42 所示。

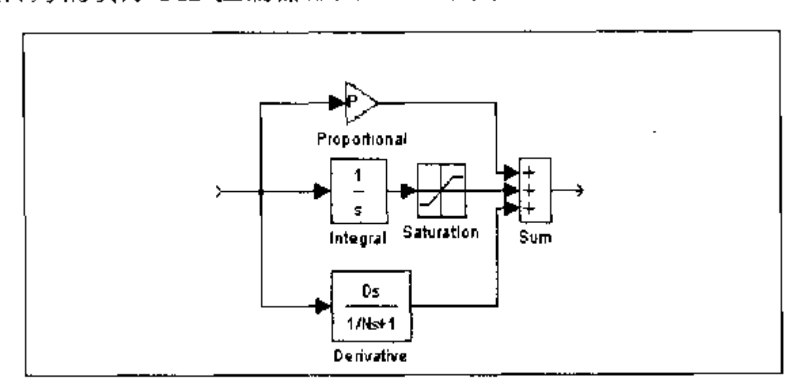


图 6－42 实际 PID 控制器简图

求解过程：本例的求解分为以下几步：

1. 构建实际 PID 非线性模型及理想 PID 模型

首先在 SIMULINK 元件库中选择非线性系统元件库，选择标有"Saturation"的 Saturation 饱和非线性元件。将其拷贝到 SIMULINK 仿真环境下。

为了简单起见，我们选择 SIMULINK 提供的理想 PID 控制器。方法是：在 SIMULINK 元件库中选择左下角的模块和工具箱元件库(Blocksets& Toolboxs)，双击其中标有"Simulink Extras"的 图标，进入其界面后再选择标有"Aditional Linear"的 图标，进入如图 6－43 界面。

选择其中标有"PID Controller"的图标，将其拷贝到 SIMULINK 仿真环境下。双击该图标，可以设置理想 PID 控制器的参数如图 6－44 所示。

有了两个主要的仿真元件后我们就可以按照图 6－45 所示的系统方框图模型从相应的元件库中选取元件了：

除了上文介绍的两个主要元件之外，图 6－45 中还有两个输入源元件库的"Step"元件，两个输出方式元件库的"Scope"元件，其余都是线性系统元件库里的，分别是"Gain"元

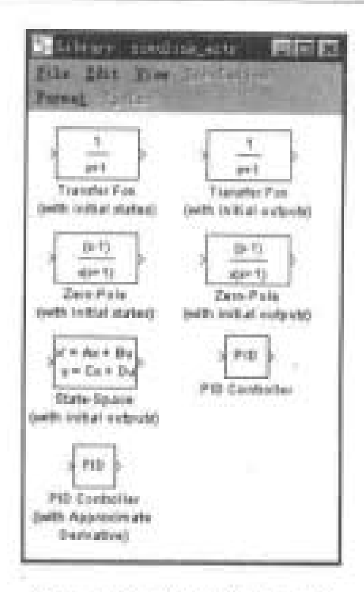


图 6-43 附加的线性元件

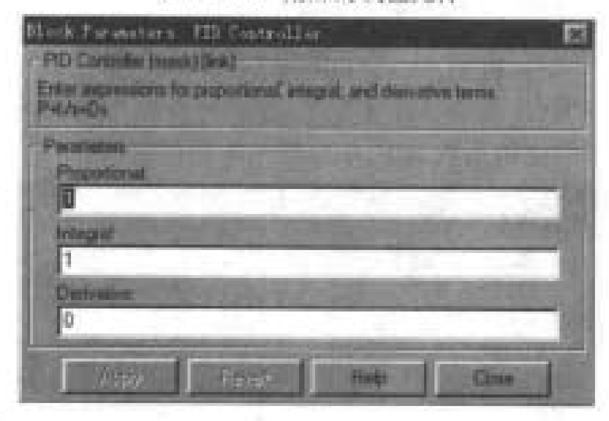


图 6-44 理想 PID 参数设置

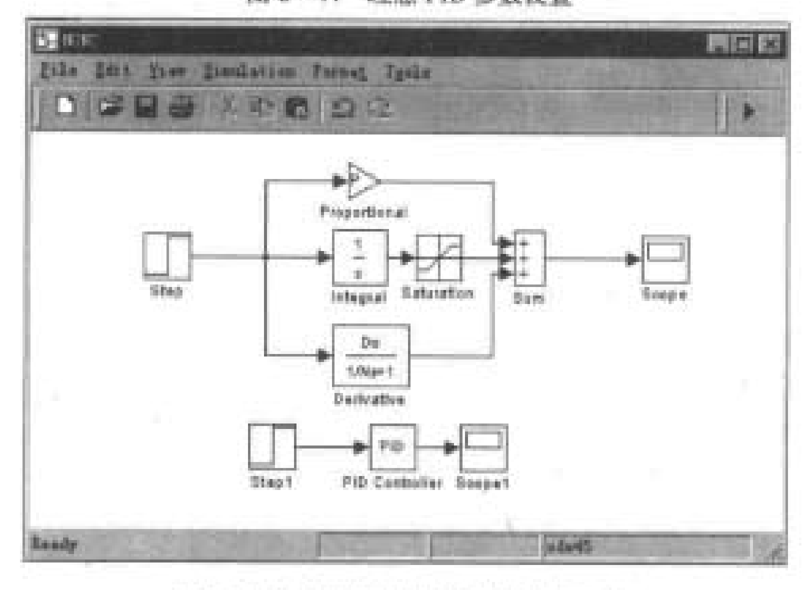


图 6-45 实际 PID 结构和理想 PID 元件

件(图中的 Proportional)、"Integrator"元件(图中的 Integral)"Transfer Fcn"元件(图中的 Derivative)和"Sum"元件。

2．设置实际PID控制器参数

首先设置饱和非线性环节。双击图6－45中标有“Saturation”的元件，有：

在图6－46中“Parameters”复选框里，“Upper limit”是非线性输出的上限，“Lower limit”是非线性输出的下限。上下限之间的部分以1比1的线性关系输出。将该非线性环节设在积分器元件之后，就是希望在上下限之间的部分保持积分器的作用，而超过上下限的部分则积分器不起作用。

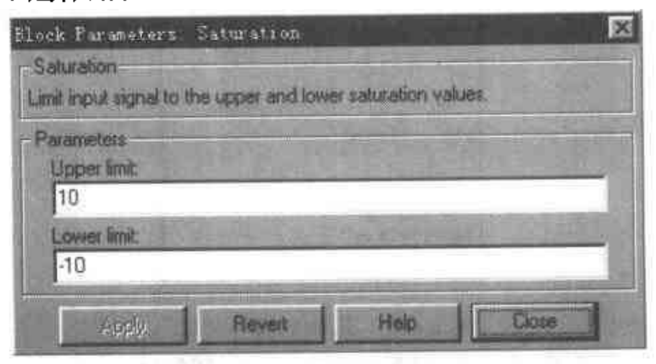


图6－46 饱和非线性元件参数设置

为了保证该实际PID控制器的可移植性，将比例器和微分器的参数都设置为MATLAB变量，其中比例器的比例系数为P，微分器的传递函数为：

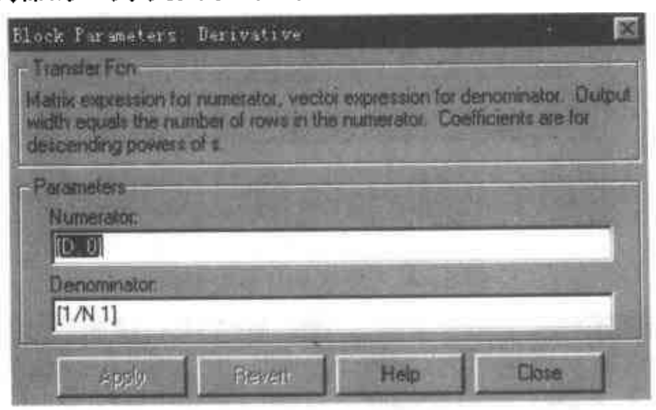


图6－47 微分器参数设置

$$D(s)=\frac{Ds}{(1/N)s+1}$$

在保证D和N均为正整数的情况下，该传递函数的最高阶分子多项式系数大于分母多项式系数，化简之后是一个微分项减去一个一阶惯性环节。这样作的好处是：一方面保证了系统的物理可实现性，单纯的微分环节很难实现，并且容易引入高频干扰；另一方面可以使微分的作用时间长一些，否则对于阶跃函数输入来说，微分环节仅仅在函数跳变的0时刻起作用，系统的输出还来不及反应。

在MATLAB Command Window下输入实际PID控制器的参数：

```
P = 10;
D = 10;
N = 1;
```

理想PID控制器的参数可以通过双击该元件来直接设计，按照实际PID控制器的参

数相应的令 P = 10，I = 1，D = 10。

3. 系统仿真

首先在仿真环境的“Simulation”菜单“Parameters”选项中将仿真结束时间设置为 12s，将仿真的相对精度设为 1e - 5。然后选择“Simulation”菜单“Start”选项或单击 按钮，开始系统仿真。

等到仿真结束后，单击图 6 - 45 的“Scope”元件，即可看到实际 PID 控制器的阶跃响应曲线如图 6 - 48 所示。

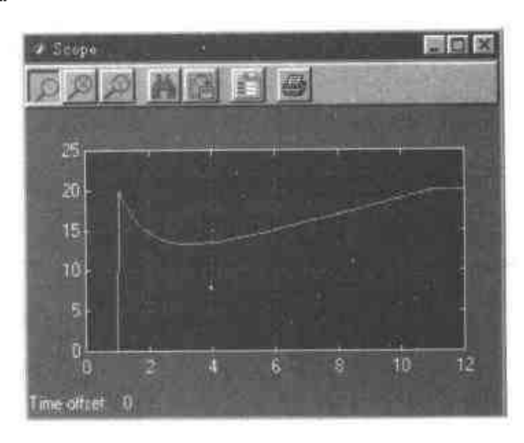


图 6 - 48 实际 PID 控制器的阶跃响应曲线

单击图 6 - 45 的“Scope1”元件（注意，不是“Scope”），即可看到理想 PID 控制器的阶跃响应曲线如图 6 - 49 所示。

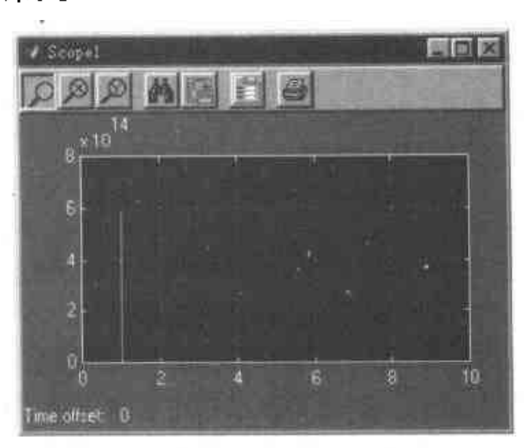


图 6 - 49 理想 PID 控制器的阶跃响应曲线

首先，从图 6 - 49 理想 PID 控制器的阶跃响应曲线中可以看到，在 T = 1 仿真开始时刻有一根很高的直线，数量级约为 10 的 14 次方，这是对阶跃函数跳变沿微分得到的结果。其后的曲线基本为零（并不是真正为零，只是与 10 的 14 次方比较，数量级相差太远而已）。显然，用有这样阶跃响应特性的校正装置是无法真正投入工业实际应用中去的，在物理上也不可能实现这种特性。

然后，我们对照图 6 - 48 的实际 PID 控制器的阶跃响应曲线，发现 T = 1 仿真开始时刻系统的输出值为 20，并且微分作用也不是像图 6 - 49 那样立即消失，而是逐渐变小，大

约在 3s 左右达到最低点。经过该最低点后积分器开始起主导作用，因此到最高点之前的这段曲线与阶跃函数的积分曲线基本相同，近似是斜率为 1 的直线(因为横坐标和纵坐标刻度不同，看起来斜率不为 1)。达到最高点后积分器饱和，不再起作用，系统的输出由比例环节控制。因此系统的输出值是积分器的输出上限 10 加上比例系数 10 乘以阶跃函数幅值 1，最后值为 20。

因此，实际 PID 控制器在响应的不同阶段不同的环节起作用，有利于设计者分析控制性能指标和参数整定。此外，饱和非线性环节在这里起了非常重要的作用，它使得 PID 控制器与实际的控制器执行环节结合更加紧密，对完成实际 PID 控制器的设计和仿真起着至关重要的作用。

小结：本例介绍了饱和非线性元件在 PID 控制器的设计和仿真中的应用，详细讲述了有关非线性系统元件的选取和使用。

非线性系统元件库中还有许多其他的非线性元件，其具体的使用方法都和本例类似。通过本例我们可以看到，使用 SIMULINK 进行非线性系统仿真可以摆脱列写繁琐的系统非线性方程，仅需选择相应的元件搭建方框图即可，并且还可以确保准确性，这是它的优点。其缺点就是许多非常成熟的分析非线性系统的方法例如描述函数法、波波大法等等还不能直接应用，必须与 MATLAB 函数结合起来才能进行分析和处理。在 MATLAB 环境下，我们可以调用非线性系统工具箱来解决这些问题。

多数情况下，控制系统中存在着非线性因素，对系统的控制性能会产生不利的影响。但在控制系统中恰当的加入特定的非线性因素，有时却能使控制性能得到改善。这种人为的加入系统内部的非线性特性称为非线性校正装置。对某些系统采用简单的非线性校正，甚至能够受到线性校正不能比拟的结果。请看下例：

[例 6.5] 某控制系统的传递函数如下。分别接入速度反馈和死区非线性反馈，进行仿真，并比较其结果：

$$G(s)=\frac{K}{s(s+1)}$$

结果：不接入速度反馈和非线性反馈时，系统的响应曲线振荡比较剧烈，超调量比较大。接入速度反馈后系统没有超调量，但响应速度很慢。接入速度反馈和非线性反馈后，各项指标都得到了改善。具体的结果请看后面的曲线。

分析：所谓速度反馈，就是将积分器之前的输出作为反馈信号引回输入端，而不是将系统的输出作为反馈信号引回输入端(这是假设系统的输出是位移而得到的说法，位移的微分就是速度)。这种反馈常用在电机控制或机械系统控制方面，可以有效降低系统的超调量。死区非线性的特性和饱和非线性的特性正好相反，将死区非线性元件和速度反馈配合使用可以收到很好的效果。

求解过程：

本例的求解分为以下几步：

1. 原始系统仿真

在 SIMULINK 环境下构建原始系统的方框图，如图 6－50 所示。

图中各元件的选取方法在前边都曾经叙述过。本例选取 K = 10，选择"Simulation"菜单"Start"选项，进行仿真，如图 6－51 所示。

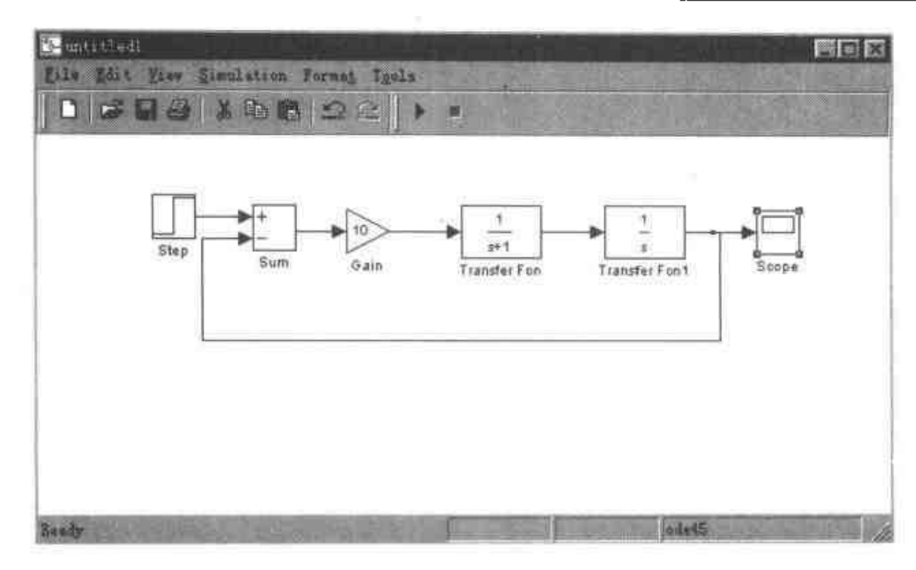


图 6-50　未加反馈的系统方框图

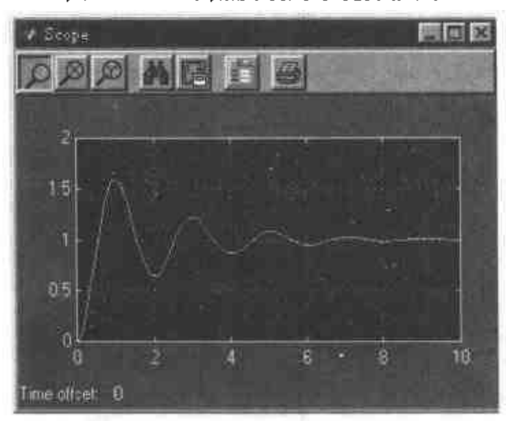


图 6-51　原始系统仿真曲线

可以看到,未接入速度反馈前系统的超调量大于 50%,并且振荡次数也比较多。但因为存在积分环节,所以系统是静态无差的。并且,系统的上升时间也比较快。不过因为振荡比较剧烈,所以过渡过程时间比较大。

2. 接入速度反馈系统的仿真

在 SIMULINK 环境下构建接入速度反馈后系统的方框图如图 6-52 所示。

速度反馈就是图中增益元件"Vfeedback"。本例选取 Kv = 3,选择"Simulation"菜单"Start"选项,进行仿真如图 6-53 所示。

从图中可以明显看出,系统响应曲线变成了过阻尼型,没有了超调量,可见速度反馈可以有效的降低系统的超调量。但是系统的上升时间变得很慢,快速性没有得到保障。因此还要寻找别的校正手段。

3. 接入死区非线性反馈系统的仿真

在 SIMULINK 环境下构建接入死区非线性反馈后系统的方框图如图 6-54 所示。

其中标有"Dead Zone"的死区非线性元件是从非线性系统元件库中选择的。双击此图标,可进行参数设置,如图 6-55 所示。

其中"Start of dead zone"是死区的起点,"End of dead zone"是死区的终点,二者之间

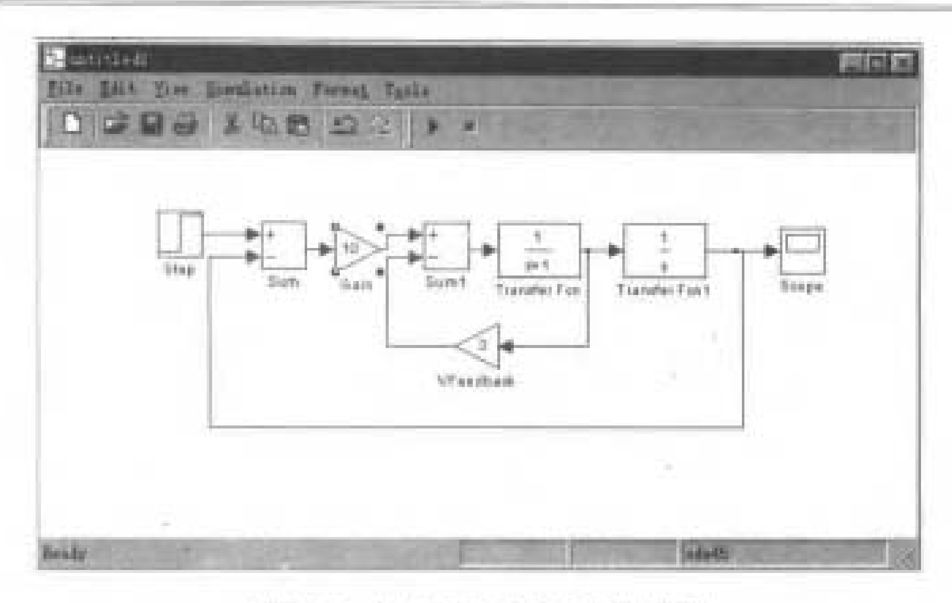


图 6-52　接入速度反馈后系统的方框图

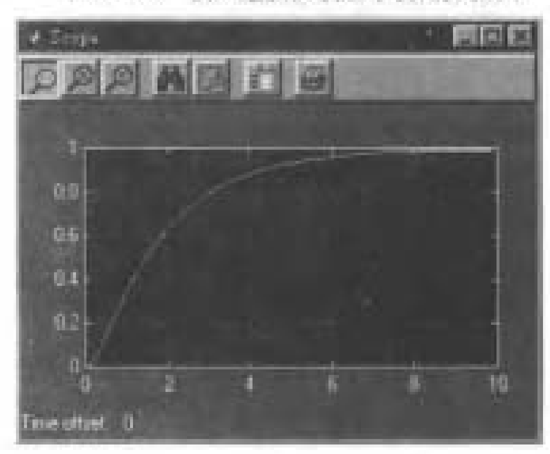


图 6-53　接入速度反馈后系统的阶跃响应曲线

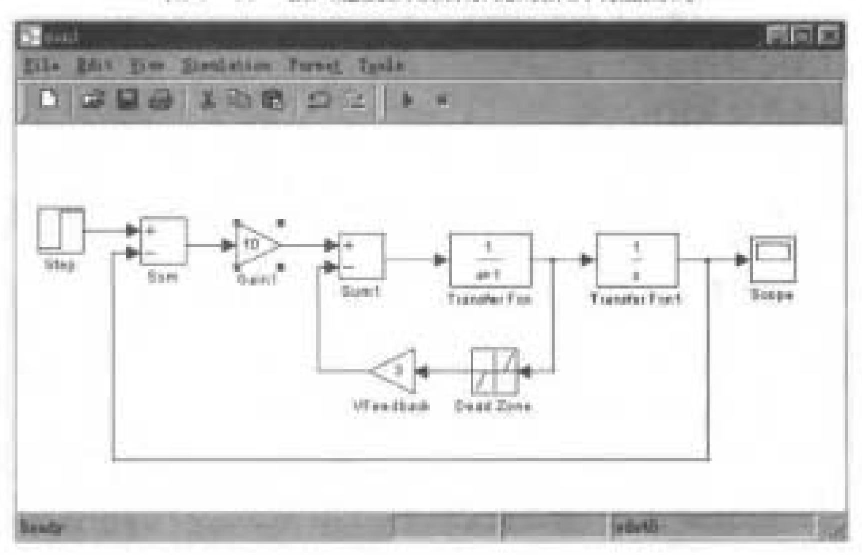


图 6-54　接入死区非线性反馈后系统的方框图

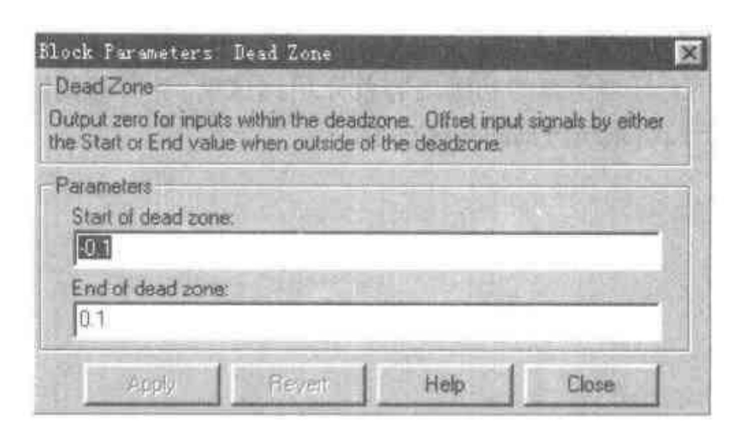


图 6-55　死区非线性元件参数设置

的距离是死区的宽度。死区外是斜率为 1 的线性函数。如果希望斜率不为 1，可以在死区元件的输入端和输出端接增益元件，调整到希望的斜率。本例设置死区的起点为 -0.1，死区的终点为 0.1。选择“Simulation”菜单“Start”选项，进行仿真如图 6-56 所示。

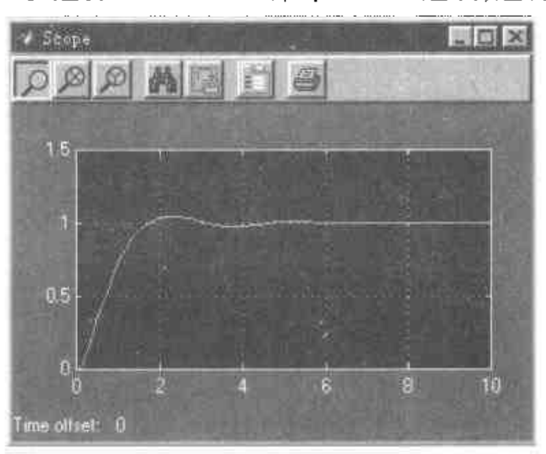


图 6-56　接入死区非线性反馈后系统的阶跃响应曲线

对比图 6-51、图 6-53 和图 6-56 可以看出，接入死区非线性校正装置后，系统的上升时间大约 1s，超调量小于 5%，过渡过程时间不超过 3s，综合了输出反馈和速度反馈的优点而补偿了它们的缺点。

小结：本例介绍了一个简单的非线性反馈校正的例子，目的是使读者进一步了解 SIMULINK 环境下非线性系统仿真的思路和方法。从结果来看，此非线性反馈的效果还是很理想的。事实上，方便快捷的构建和分析非线性系统模型，也是 SIMULINK 环境和 MATLAB 的优势之一。

6.3　小　　结

本章我们主要介绍了 SIMULINK 仿真环境的一些基本概念和初步的用法，包括如何在 SIMULINK 环境下进行控制系统仿真、SIMULINK 环境概述、如何在 SIMULINK 环境实现控制系统的传递函数模型、状态方程模型和非线性模型的描述。本章的重点是帮助没有接触过 SIMULINK 环境的读者建立起一些基本概念，能够在 SIMULINK 环境

下进行简单的控制系统仿真与设计。

有关 SIMULINK 环境仿真的初级内容就介绍到这里，下一章我们将重点介绍一些 SIMULINK 环境下控制系统仿真的高级技术。

第七章　SIMULINK 高级技术

在上一章的介绍中我们了解到,利用 SIMULINK 环境可以非常轻松地搭建出复杂控制系统的模型,同时还可以很方便地绘制出系统的仿真曲线,并且可以将仿真结果输出到 MATLAB 的工作空间中去。事实上,SIMULINK 还有许多特殊的功能可以使我们更加方便地进行控制系统设计和仿真。例如通过 LTI Viewer 可以利用图形分析线性定常系统;使用 Group 方法可以将拥有某些通用功能的公共部分构成 SIMULINK 模块,方便以后的应用;用户还可以通过封装的办法构成一个 SIMULINK 子系统,并按照一定的格式设置子系统的输入输出参数。

7.1　线性定常系统仿真 LTI Viewer

SIMULINK 提供了一个线性定常系统的可视化仿真环境——LTI Viewer。在该环境下可以从 MATLAB Workspace 或 SIMULINK 中输入系统的模型,根据不同的要求绘制相应的曲线,例如阶跃响应曲线、脉冲响应曲线、Bode 图、Nyquist 图和 Nichols 图等等。从不同的曲线中,还可以求取各种性能指标参数,例如过渡过程时间、峰值点、增益裕量、相角裕量、谐振峰等等。我们在前几章里分析的线性定常系统的各种曲线和性能指标几乎都可以在这里找到。

下面我们首先通过一个简单的例子使读者对 LTI Viewer 环境有一个初步的认识,然后再给出一个具体的应用详细介绍 LTI Viewer 环境的各项功能。

7.1.1　LTI Viewer 环境

请看下面这个例子,我们一边求解,一边介绍 LTI Viewer 环境的使用方法。

[例 7.1]　某控制系统的开环传递函数如下所示。试在 LTI Viewer 环境下求解其闭环阶跃响应曲线。

$$G(s)=\frac{1}{s(s+1)}$$

结果:系统的闭环传递函数系数为:

```
nume =   0     0     1
dene =   1     1     1
```

对应传递函数为:　$$G(s)=\frac{1}{s^2+s+1}$$

系统的闭环阶跃响应曲线见图 7-1。

分析:由于本例的传递函数模型比较简单,所以我们在 MATLAB Workspace 中直接设置系统的模型的参数。系统的闭环传递函数可以用 MATLAB 提供的 cloop( )函数求

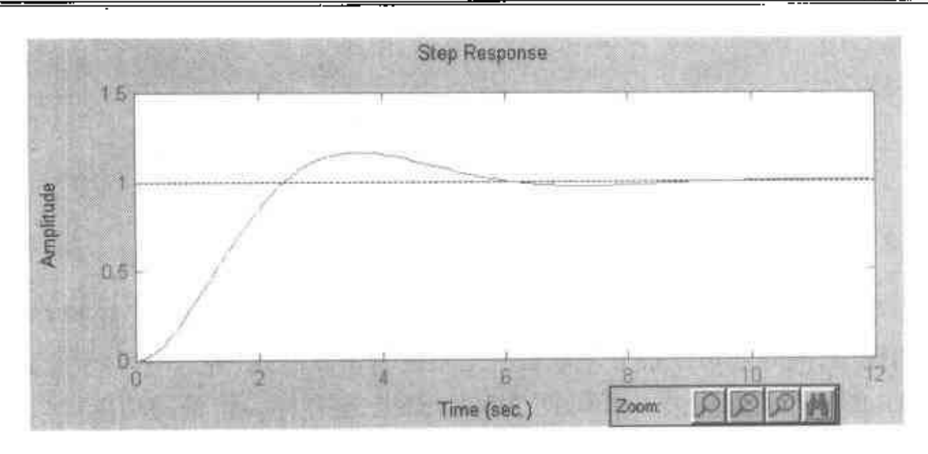


图 7-1 LTI Viewer 下系统闭环阶跃响应曲线

解，阶跃响应曲线可以在 LTI Viewer 下绘制。

求解过程：本例的求解分为以下几步：

1. 在 Workspace 下设置系统模型

在 MATLAB Command Window 下键入下列语句：

```
num = 1;
den = [1 1 0];
[numc,denc] = cloop(num,den, - 1);
sys = tf(numc,denc);
```

系统模型的抽象描述名称为“sys”。

2. 进入 LTI Viewer

在 SIMULINK 仿真界面下选择“Tools”菜单的“Linear Analysis”选项，即可进入 LTI Viewer 可视化仿真环境，如图 7-2 所示。

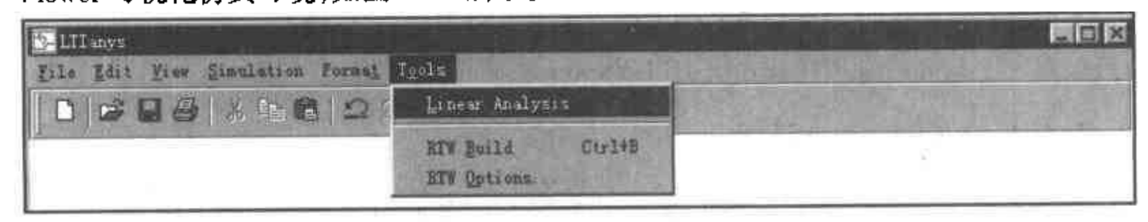


图 7-2 进入 LTI Viewer 环境

进入 LTI Viewer 环境后首先看到的是模型输入点和输出点设置界面。这是基于 SIMULINK 方框图的 LTI Viewer 工具，与本例无关，我们将在下一小节介绍。然后看到的就是 LTI Viewer 仿真主界面，如图 7-3 所示。

该界面上半部分是绘制系统响应曲线的区域，下半部分是选择系统模型和响应曲线类型的区域。下面所有操作步骤和曲线的绘制都在此界面下进行。

3、绘制仿真曲线

在图 7-3 的左下方有一个标有“System”的复选框，其功能是选择输入的系统名称。在第一步中我们已经在 Workspace 里设置了线性系统“sys”，因此用鼠标单击“Refresh”按钮，有：

双击 Workspace 对话框中的“sys”或选中“sys”后单击“Select”按钮，Selected 对话框中

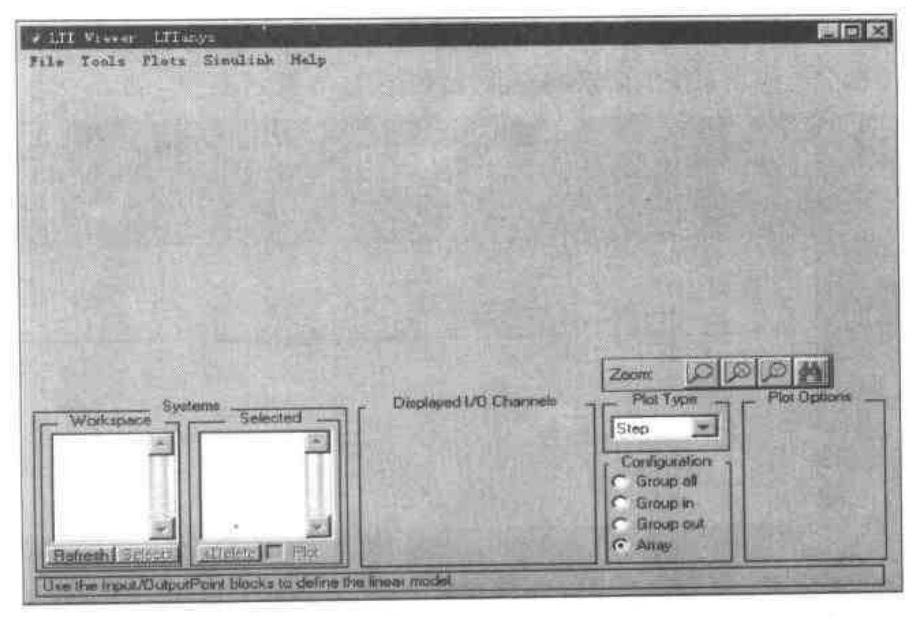


图 7-3 LTI Viewer 仿真界面

就会出现如图 7-4 所示的“+sys(red)”字样，表示该系统已经被选中了。“red”表示为该系统绘制的曲线都是红颜色的。

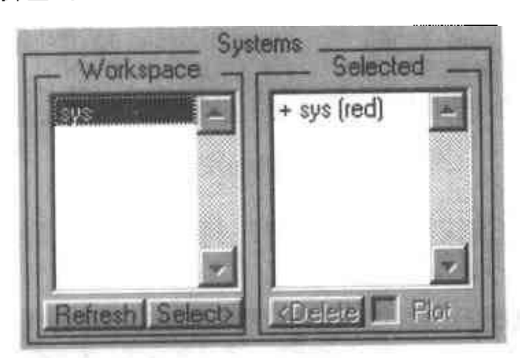


图 7-4 选择系统模型

在图 7-3 的右下方有一个标有“Plot Type”的复选框，其功能是选择需绘制曲线的类型。打开下拉菜单，可以看到如图 7-5 所示的界面。

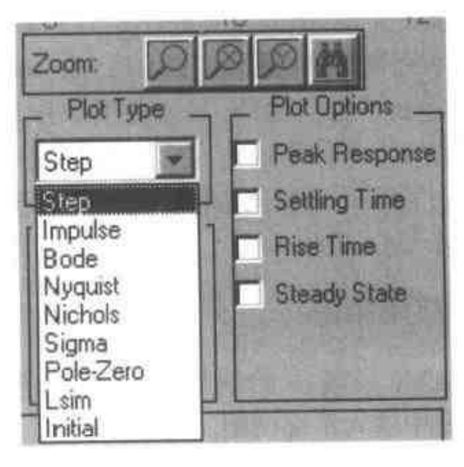


图 7-5 选择响应曲线类型

本例是要求绘制阶跃响应曲线,因此选择"Step"选项,此时系统就在 LTI Viewer 主界面的上半部分绘制出系统的阶跃响应曲线,如图 7-6 所示。

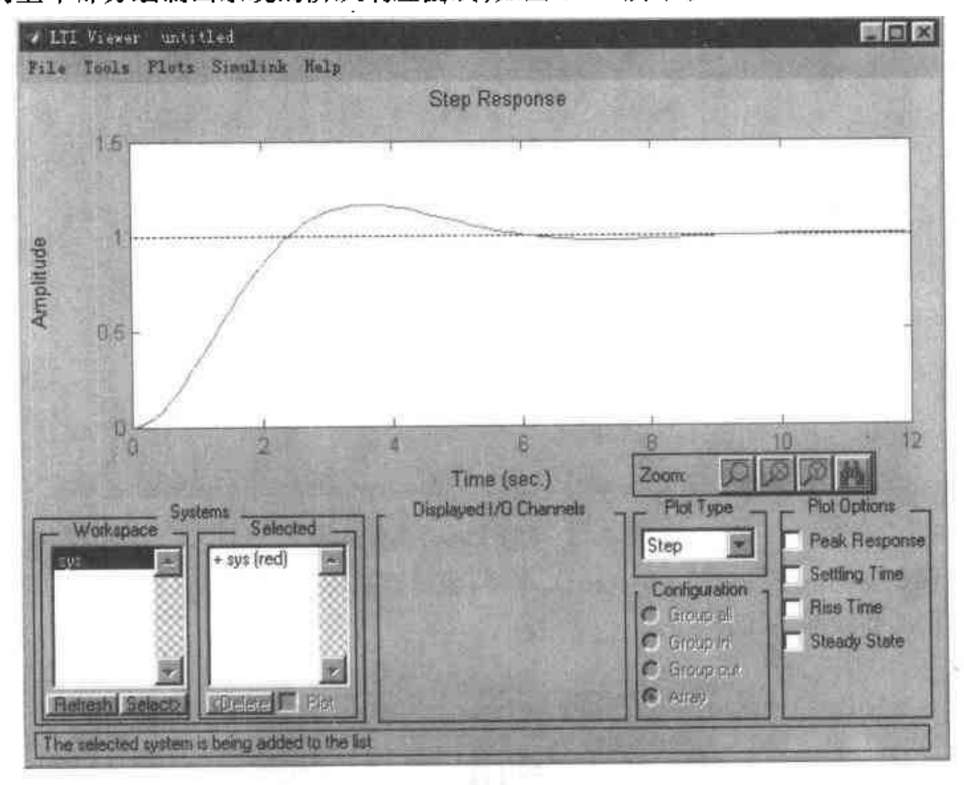


图 7-6 闭环系统的阶跃响应曲线

我们还可以给该响应曲线加网格,方法是:打开图 7-6 所示界面的"Plots"菜单,选择"Grid on"选项。如图 7-7 所示。



图 7-7 给响应曲线加网格

小结:本例的系统模型非常简单,因此重点并不是分析系统的性能,而是帮助读者熟悉 LTI Viewer 环境。通过本例我们可以看出,在 LTI Viewer 环境下进行控制系统仿真,除了模型参数需要在 MATLAB 下建立之外,其余各项工作只需点击鼠标即可完成。并且,LTI Viewer 还可以从 SIMULINK 环境输入系统的方框图模型。这样一来,从系统建模到系统的分析和仿真,全过程都是可视化操作。尤其是对于大型控制系统来说,非常直观、方便。这也是我们下一小节介绍的内容。

7.1.2 LTI Viewer 应用举例

LTI Viewer 是 SIMULINK 环境下的线性定常系统仿真工具,因此当然可以从 SIMULINK 下的系统方框图中输入系统模型。请看下例:

[例 7.2] 系统的状态方程描述如下。试搭建其方框图模型,应用 LTI Viewer 分析其系统性能,并给出相应的状态反馈矩阵改善其动态性能。

$$A=\begin{bmatrix}0 & 1 & 0\\ 980 & 0 & -2.8\\ 0 & 0 & 100\end{bmatrix}\qquad B=\begin{bmatrix}0\\0\\100\end{bmatrix}$$

$$C=[1\quad 0\quad 0]\qquad D=0$$

结果:详见下边解题步骤中的具体步骤。

分析:在上一章里我们曾经提到,SIMULINK 环境下的状态空间模型"State - space"元件只有一个输出端,无法提供状态变量的数据。因此,有必要将系统结构转换为能够提供状态反馈的形式。考虑到第二章曾经介绍过的系统的第一可控规范型形式,如图 7 - 8 所示,各个积分器后的输出就是系统的状态变量,可以直接引出形成状态反馈。因此,我们首先将原系统的状态方程化为第一可控规范型的形式,将相应的系数填入 SIMULINK 构建的系统第一可控规范型方框图里边去,然后再设计和调整系统的状态反馈,就可以完成本例的要求。

求解过程:

本例的求解分为以下几步

1. 求解系统第一可控规范型

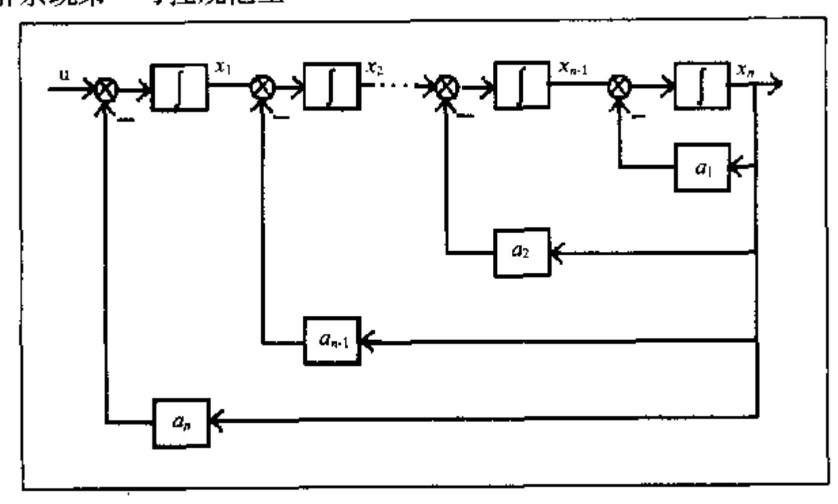


图 7 - 8　系统的第一可控规范型方框图

关于系统可控规范型的定义和求解算法读者可参看本书第二章,这里就不再赘述了。给出本例的代码,读者在 MATLAB Editor/Debugger 下输入存盘后调试运行即可。需要注意的是,本例结束之前,不要清除内存变量。

```
%系统的状态方程模型
A =   [0        1        0
      980       0      -2.8
        0       0      -100];
B = [0
       0
          100];
C = [1  0  0];
%系统阶次
```

```
n = length(A);
%状态可控性矩阵
Q = zeros(n);
Q(:,1) = B;
%求解状态可控性矩阵
for i = 2:n
    Q(:,i) = A * Q(:,i-1);
end
%判断系统是否可控
m = rank(Q);
if m = = n
%求解系统第一可控规范型
    Ac1 = inv(Q) * A * Q;
    Bc1 = inv(Q) * B;
    Cc1 = C * Q;
    disp('System is Controllable.');
    disp('System First Controllable Canonnical Form is:');
    Ac1
    Bc1
    Cc1
    disp('The Transformation Matrix is:');
    Q
else
%若系统不可控,则显示警告信息
    disp('System State Variables cannot be totally controlled');
    disp('The rank of System Controllable Matrix is:');
    m
end
```

2. 搭建系统方框图

根据图 7－8 的系统第一可控规范型方框图,进入 SIMULINK 后在其仿真环境下搭建如下的系统方框图,如图 7－9 所示。

本方框图选用了线性系统元件库中的"Gain"元件三个、"Integrator"元件三个、"Sum"元件三个、"Matrix Gain"元件一个;连接装置元件库的"Mux"元件一个。和以前的方框图不同,本图在系统的输入端和输出端没有使用"Step"元件和"Scope"元件,而是使用了"Input Point"块和"Output Point"块。这是 LTI Viewer 提供的元件,其功能是在 LTI Viewer 仿真界面和 SIMULINK 系统模型之间传递信息。当然,如果希望在 SIMULINK 仿真环境下绘制图形,就需要选用"Scope"元件或别的输出元件。关于这两个元件的选取和设置,我们将在下一步作出具体解释。

为了便于查找和调试,我们在图中各个关键的地方都作了标记。例如输入点 u,三个积分器后面输出的系统状态向量 x1、x2 和 x3,以及系统的输出量 y 等等。其方法就是用鼠标双击相应的位置,在弹出的文本框中填入相应的文字即可。图 7－10 是基于系统第

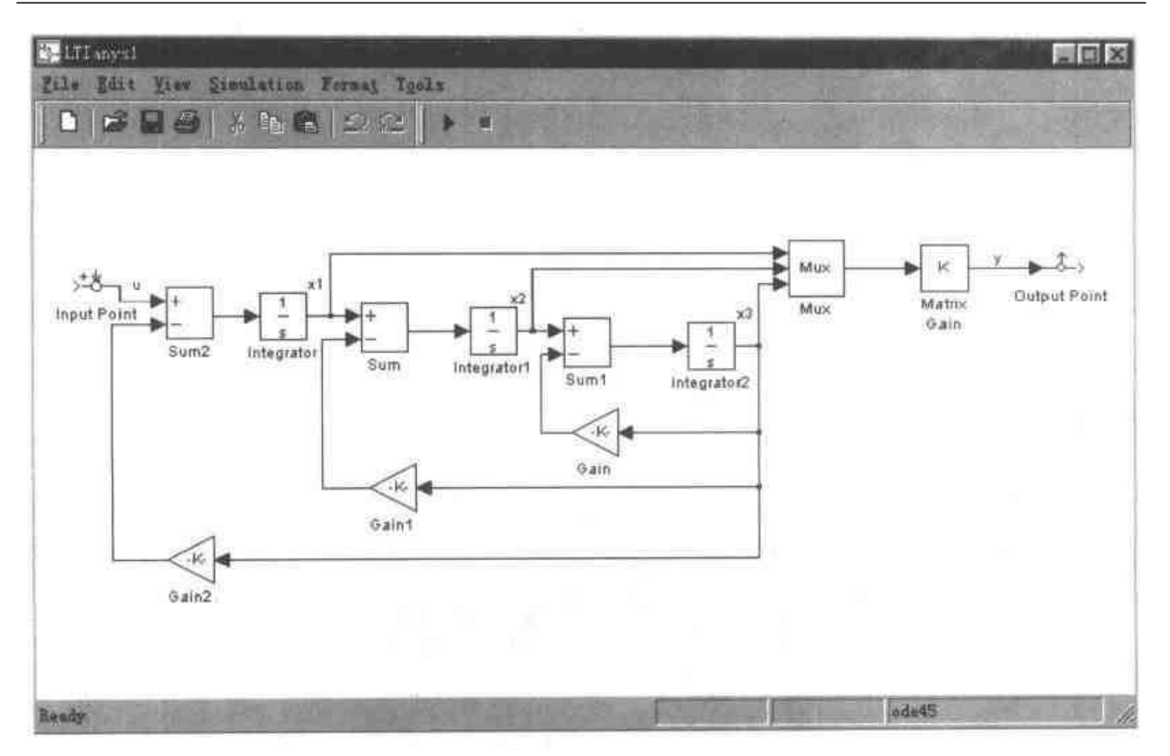


图 7-9 不含状态反馈的系统方框图

一可控规范型的方框图，因此，要用到上一步求解的系统第一可控规范型的参数矩阵。例如三个增益元件“Gain”、“Gain1”和“Gain2”，对应第一可控规范型 A 矩阵的最后一列元素 a1、a2 和 a3(此关系的求解和证明详见第二章)，因此双击该元件，填入相应的变量名称：



图 7-10 Gain 元件参数设置

这是第一个元件“Gain”的参数，相应的在其他两个增益元件的参数设置对话框中填入“-Ac1(2,3)”和“-Ac1(1,3)”。

图 7-9 中“Mux”元件的作用是将三个标量形式状态变量合成一列向量输出，因此将输入的个数设为 3(设置方法可参看第六章)。

图 7-9 中“Matrix Gain”元件的作用是将输入的状态向量乘以第一可控规范型的 C 矩阵，得到系统的输出量 y。因此，双击该元件，将其增益矩阵设置为“Cc1”，如图 7-11 所示。

为了能够在 LTI Viewer 下正确的找到该系统方框图，需要给它起一个文件名并存盘。我们将此方框图的名称定为“LTIanys1.mdl”。

到此为止，系统原始方框图的搭建就基本完成了，下面我们进入 LTI Viewer 设置输



图 7－11　Matrix Gain 元件参数设置

入点和输出点，并进行仿真。

3．原始系统仿真

在图 7－9　SIMULINK 仿真界面下选择“Tools”菜单的“Linear Analysis”选项，进入 LTI Viewer 可视化仿真环境，如图 7－12 所示。

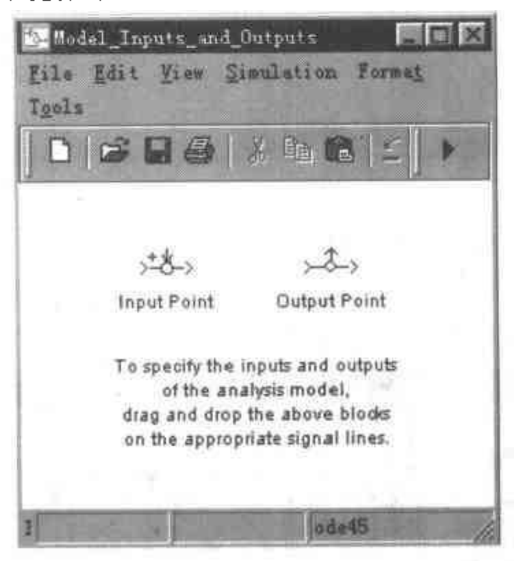


图 7－12　系统输入输出设置界面

这是 LTI Viewer 与 SIMULINK 系统方框图的接口，系统的仿真信息就通过“Input Point”块和“Output Point”块这两个元件在两个界面之间传递。具体输入函数的类型和输出方式需要在 LTI Viewer 仿真界面下设置，目前只需将这两个元件拷贝到 SIMULINK 仿真界面下，并放置到图 7－9 所示的位置即可。

前面我们介绍过如何把 MATLAB Workspace 中的系统描述引入到 LTI Viewer 下，对于本例的 SIMULINK 系统方框图，就需要打开 LTI Viewer 仿真界面的“Simulink”菜单，选择“Get Linearized Model”选项，如图 7－13 所示。



图 7－13　从 SIMULINK 中引入系统描述

此时，LTI Viewer 仿真界面左下角的"System"复选框中就会显示出本例的系统方框图名称和曲线颜色，如图 7－14 所示。

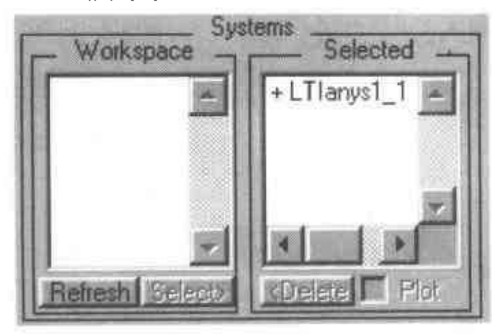


图 7－14　输入系统模型

LTI Viewer 缺省的仿真曲线类型就是阶跃响应曲线，因此，不必从"Plot Type"下拉菜单中选择曲线类型，直接就可以得到系统的阶跃响应曲线：

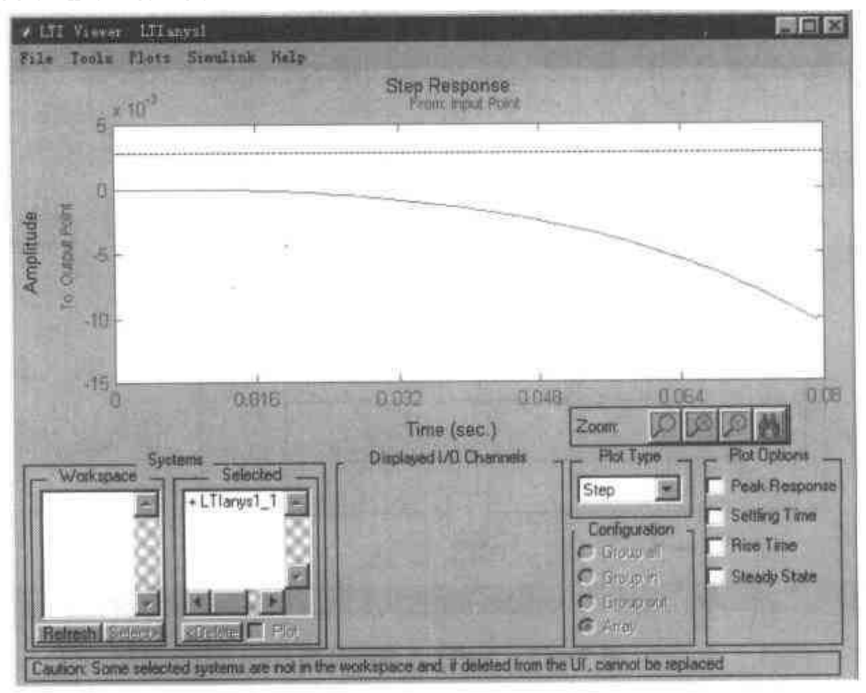


图 7－15　系统的阶跃响应曲线

从图 7－15 系统的阶跃响应曲线类型来看，系统存在不稳定的极点，曲线最终趋于负无穷。这样的系统必须进行校正，否则是不能投入实际应用的。关于各种校正的算法这里就不再介绍了，读者可以参看本书相关的章节。我们这里选用状态反馈进行极点配置，将系统的三个极点都配置到系统的左半平面，并适当调整主导极点的位置，使系统的性能指标满足实际需要。当然，我们也可以在 LTI Viewer 环境下绘制系统的 Bode 图，或是应用 rltool 环境绘制系统的根轨迹图，设计系统的串联校正装置。不过综合 MATLAB 的数值计算和 LTI Viewer 的绘图优势，我们还是选用状态反馈极点配置。

4．设置系统的状态反馈矩阵和参考输入

为了增强本例方框图的可移植性，我们在 MATLAB 下计算系统的状态反馈矩阵和

参考输入倍数,然后在方框图的相应位置填入变量名。综合各性能指标因素,我们将主导极点配置在 - 10 ± 20i,另外一个非主导极点放在远离主导极点的 - 100 处。

紧接第一步的代码,输入:

```
%系统的极点
p1 = -10 + 20i;
p2 = -10 - 20i;
p3 = -100;
P=[p1 p2 p3];
%计算状态反馈矩阵
K = place(Ac1,Bc1,P);
%计算参考输入倍数
Nbar=rscale(Ac1,Bc1,Cc1,0,K)
```

然后重新搭建含状态反馈的系统方框图如图 7 - 16 所示。

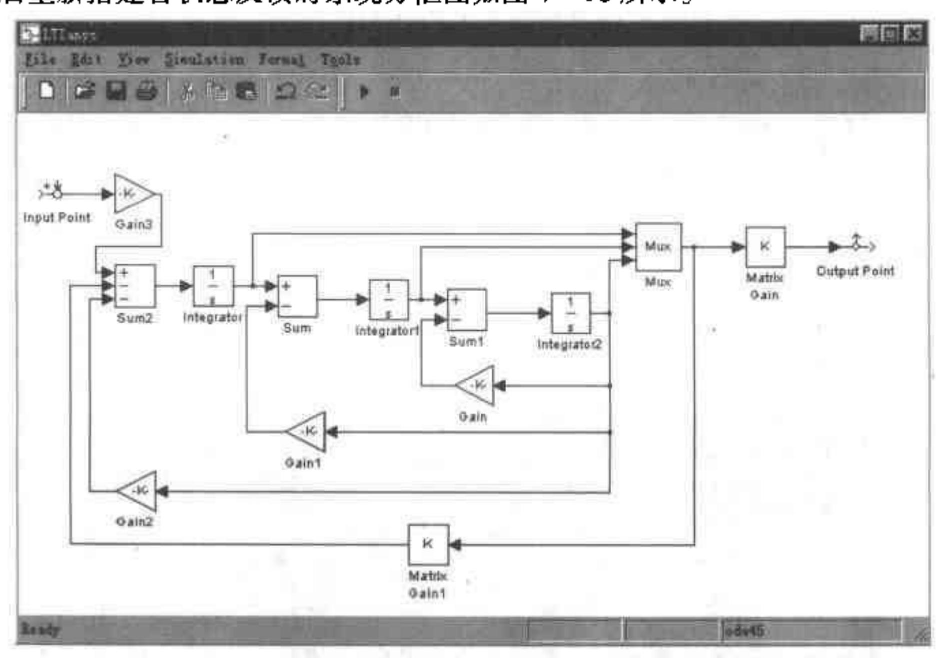


图 7 - 16 含状态反馈的系统方框图

和图 7 - 9 相比图 7 - 16 增加了两个元件,一个是与输入端与"Mux"元件输出端连接的"Matrix Gain1"元件,这就是系统的状态反馈矩阵;另一个是与输入端与"Input Point"输出端连接的"Gain3"元件,这是参考输入的倍数。相应的,在"Sum2"元件的输入端再增加一个" - "号。双击"Matrix Gain1"元件,填入前边计算状态反馈矩阵 K:

同样双击图 7 - 16 的"Gain3"元件,在弹出的对话框中与图 7 - 17 相同的位置上填入参考输入的变量名"Nbar"。

设置完毕后将此方框图另存为"LTIanys.mdl"。

5. 校正后相同仿真及分析

打开 LTI Viewer 仿真界面的"Simulink"菜单,选择"Get Linearized Model"选项。然后在仿真界面左下角的"System"复选框中选中系统方框图"LTIanys",可以得到系统的阶跃响应曲线如图 7 - 18 所示。



图 7-17 设置状态反馈矩阵

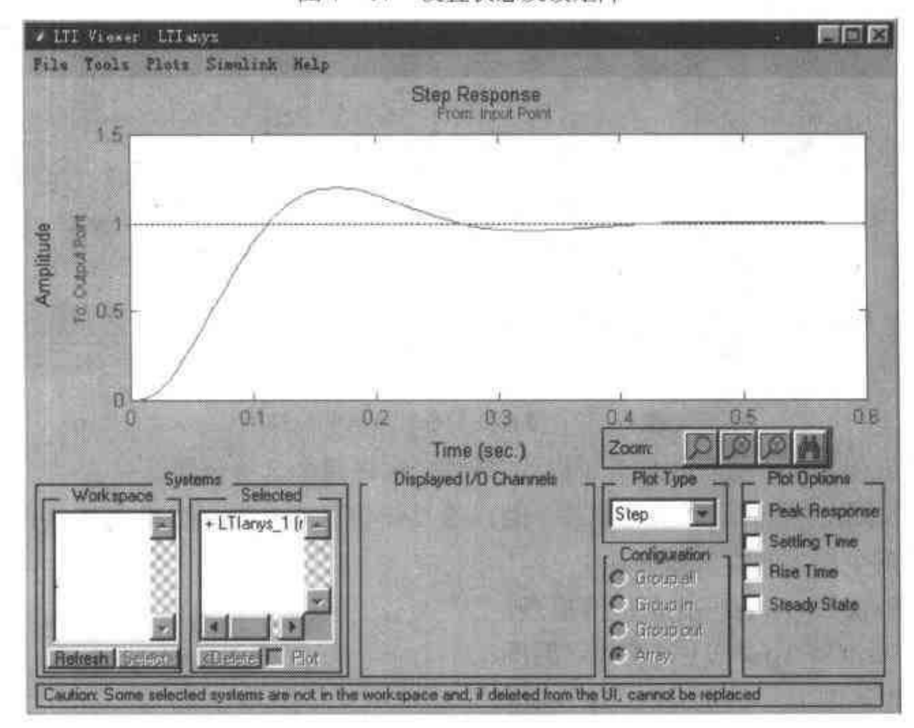


图 7-18 加入状态反馈后系统的阶跃响应曲线

从图 7-18 加入状态反馈后系统的阶跃响应曲线中可以看出，首先，系统由不稳定变为稳定，达到稳态值之前只振荡了一次；其次，系统的性能指标也比较令人满意，超调量约为 20%，过渡过程时间不超过 0.4s，上升时间也很快。可见，通过状态反馈的极点配置，基本达到了最初的控制要求。并且，系统的稳态值是 1，也就是说静态无差，这是由于加入了参考输入的缘故。

在 LTI Viewer 下进行线性定常系统仿真还有一个很大的优点，就是可以在绘制的曲线上显示我们上边提到的这些性能指标，并且不同的曲线类型可以显示不同的性能指标，不必用眼睛来估计。请看图 7-19。

从图 7-19 中曲线带圆点标记的位置我们可以读出，系统的上升时间约为 0.105s；峰值点约为 1.2，对应的超调量为 20%；系统的过渡过程时间约为 0.38s；最终稳态值为 1。用鼠标选择图 7-19 右下角的“Plot Options”复选框中的选项就可以显示这些圆点，如果每次选中一个，可以更明确的知道圆点的含义。并且，LTI Viewer 还自动将这些圆点的横坐标和纵坐标都用实线标出，我们只需按照这些与圆点相连的实线就可以读出各项控制系统性能指标的准确值，比原来通过 plot()函数绘制的曲线上凭肉眼读取性能指

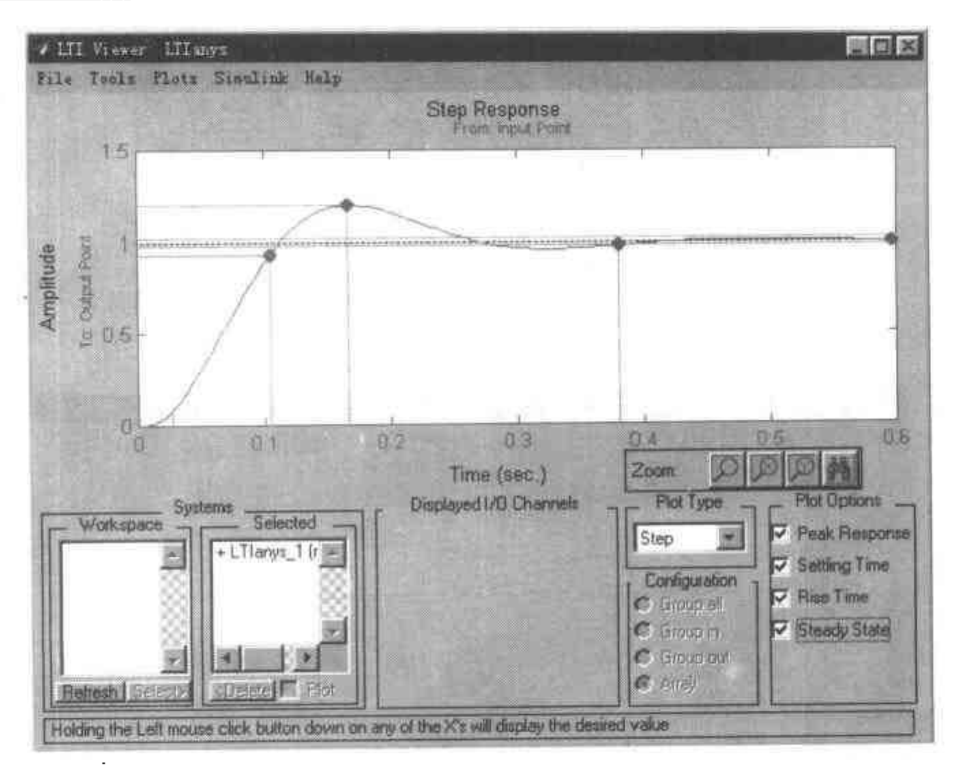


图 7－19 带标注的系统阶跃响应曲线

标数据就方便多了。这里我们为了讲述方便，一次将四个选项全部选中，因此图中有四个圆点，必须根据曲线的实际情况和各个圆点的具体位置来判断每个圆点的具体含义。该复选框中四个选项的含义如下：

（1）Peak Response——响应峰值点；

（2）Settling Time——过渡过程时间；

（3）Rise Time——上升时间；

（4）Steady State——稳态值。

由于这些值都是数值求解得出的，因此比直观用肉眼观察到的数值更加准确，并且更加方便。

前边曾经提到，LTI Viewer 仿真环境可以绘制多种类型的系统响应曲线，包括脉冲响应曲线、Bode 图、Nyquist 图、Nichols 图等等。而通过各种方式和手段反应系统的运动状态，对于线性系统分析来说是十分必要并且非常重要的。LTI Viewer 仿真环境使得这一切都以一种可视化的操作手段来完成，例如，从图 7－19 的“Plot Type”下拉菜单中选择“Bode”选项，可以得到系统的 Bode 图如图 7－20 所示。

这是闭环系统的 Bode 图，因此低频段系统的增益是 0dB，高频段斜率大约为 －3（－60dB每十倍频程），和原系统的阶次相符。相应的，频率趋于无穷时相角趋于 －270°。在图7－20 的左下角“Plot Options”中有两个选项：

（1）Gain/Phase Margin——增益/相角裕量；

（2）Peak Response——响应峰值。

选中这两个选项后可以从图中读出系统的增益裕量约为 －10dB，相角裕量约为 60°；响应的峰值在 0dB 附近。

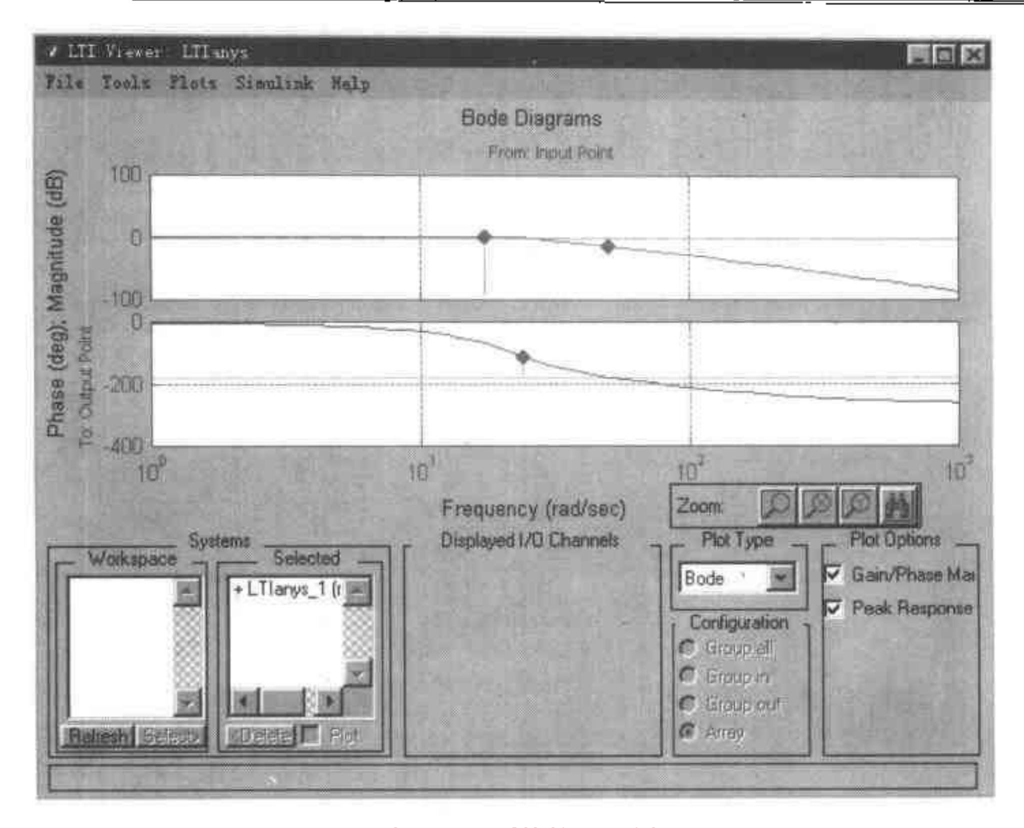


图 7－20 系统的 Bode 图

LTI Viewer 仿真环境还可以提供多输入多输出通道的仿真，例如我们在图 7－16 系统方框图的状态变量 x1 后面加入一个"Output Point"元件，如图 7－21 所示。

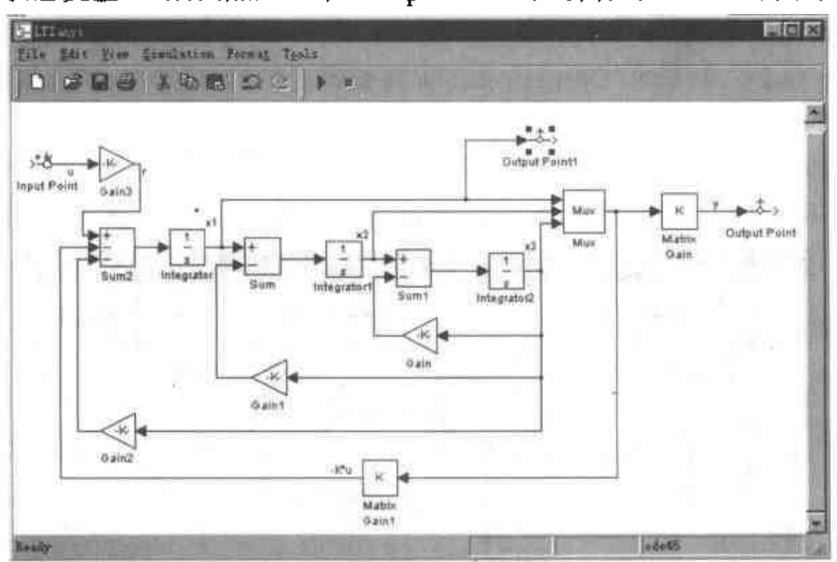


图 7－21 双输出系统方框图

按前边介绍的方法，打开 LTI Viewer 仿真界面的"Simulink"菜单，选择"Get Linearized Model"选项。然后在仿真界面左下角的"System"复选框中选中系统方框图"LTIanys"，可以得到系统的阶跃响应曲线如下：

图 7－22 上边的曲线纵坐标方向标注"To: Output point"，说明是图 7－21 中"Output

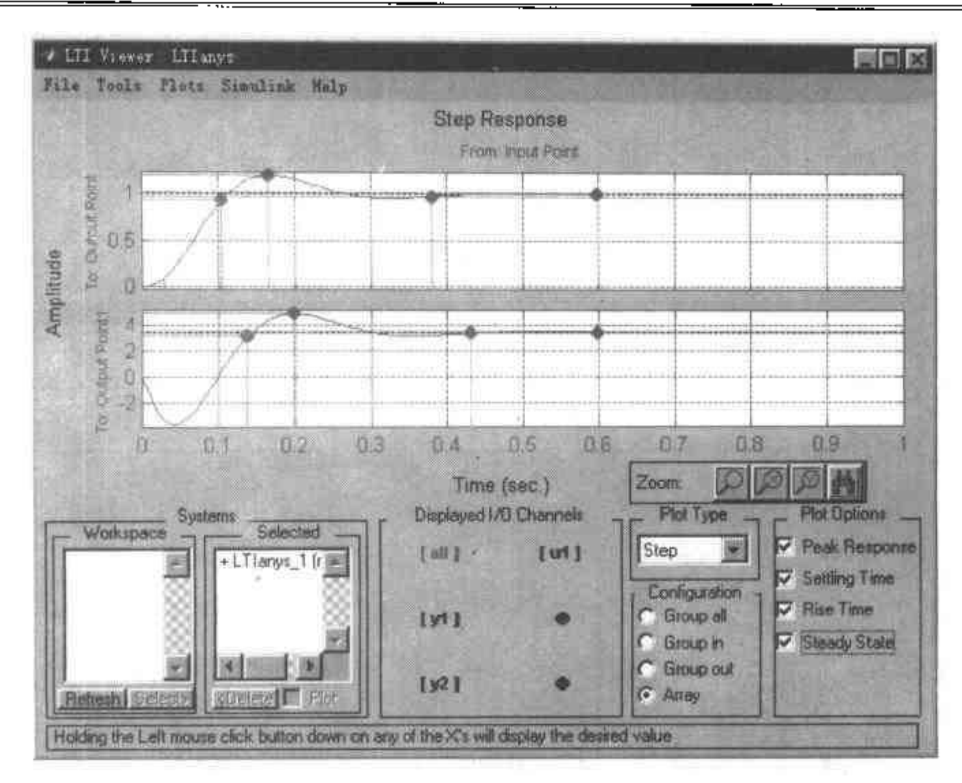


图 7－22 双输出系统阶跃响应曲线

point”元件的输出量，即系统的输出量 y；下边的曲线纵坐标方向标注“To: Output point1”，说明是图 7－21 中“Output point1”元件的输出量，即系统的状态变量 x1。在图 7－22 下半部分的中间有一个“Displayed I/O Channels”复选框，在该复选框中可以选择显示哪一个输出通道的数据。例如图 7－22 选中的是“[all]”，所以就显示全部两个通道的曲线；如果选“[y1]”，则显示“Output point”元件的输出量；如果选“[y2]”，则显示“Output point1”元件的输出量。

最后，我们再介绍一下 LTI Viewer 仿真环境的仿真参数设置。打开 LTI Viewer 仿真环境下的“Simulink”菜单，选择“Set operating point”选项，可以设置状态变量的初始值：

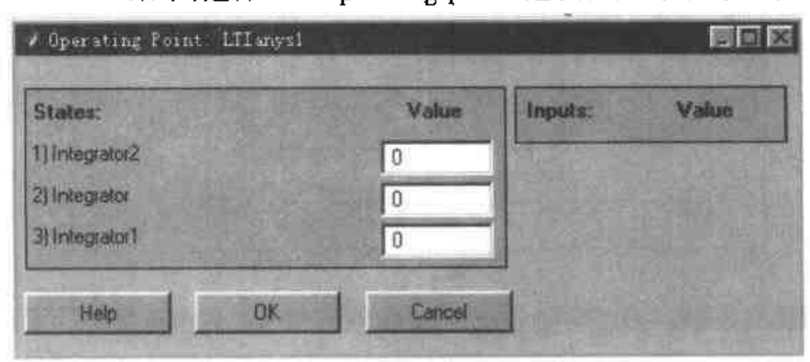


图 7－23 状态变量初始值设置界面

LTI Viewer 环境认为系统方框图的积分器输出就是状态变量，可以在图 7－23 的对话框中设置例 7.2 的三个状态变量的初始值。

打开 LTI Viewer 仿真环境下的“Tools”菜单，选择“Response Preferences”选项，可以设置仿真环境的各种参数：

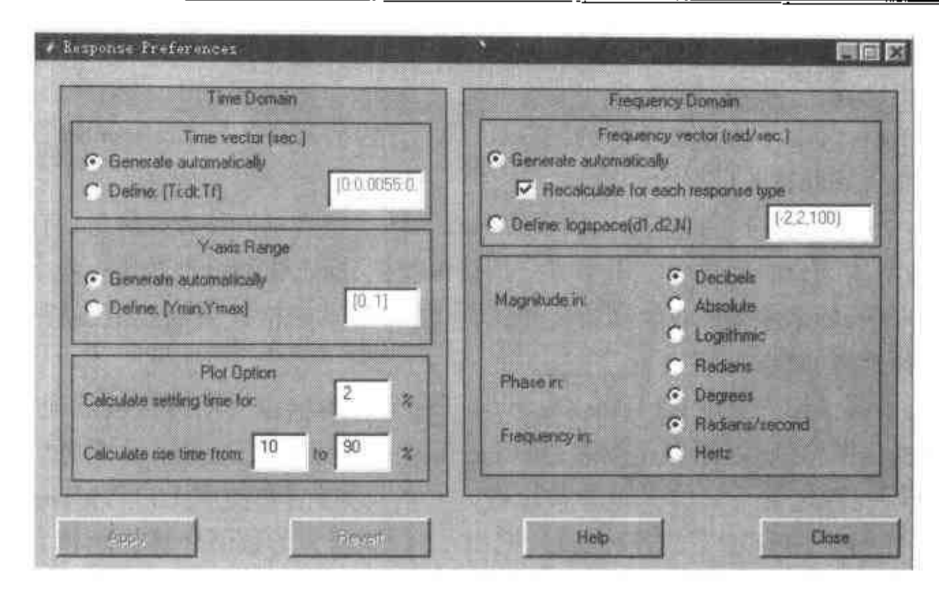


图 7－24　仿真环境参数设置界面

图 7－24 的参数设置界面分为时间域和频率域两部分。时间域参数包括仿真起止时间、步长以及坐标范围，不过对于这些参数，如果没有特殊要求一般都选取"Generate Automatically"选项，由仿真环境自动产生。从时间域的第三个复选框中可以看出仿真环境对于过渡过程时间和上升时间的缺省设置，分别是进入系统稳态值 2% 区域内的时间和从稳态值的 10% 上升到 90% 所用的时间。当然，读者也可以键入自己设定的数值。频率

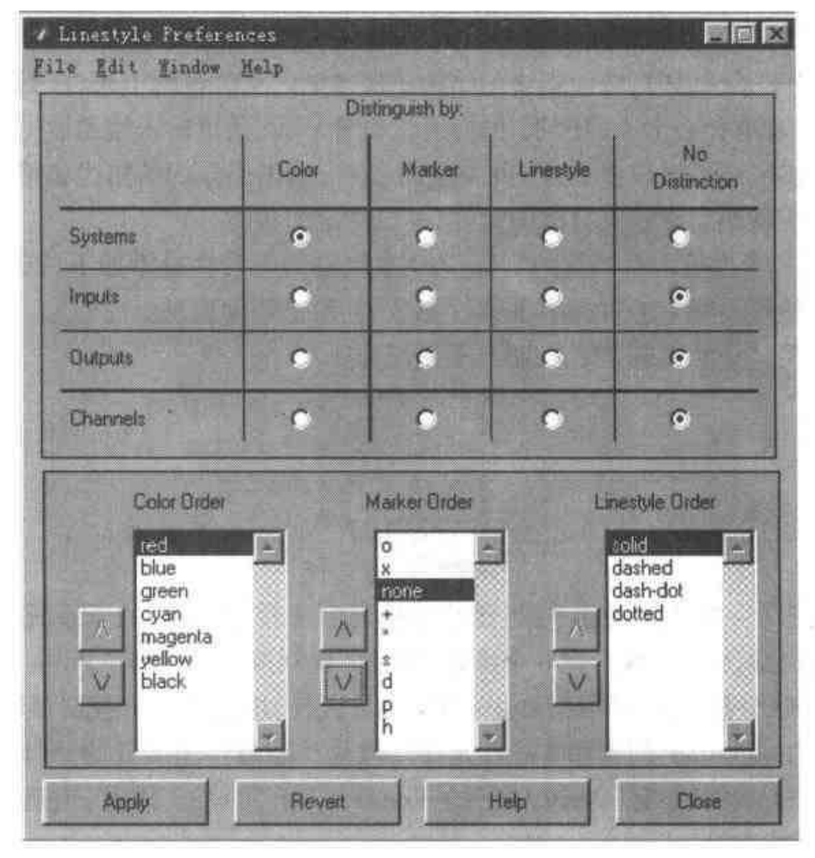


图 7－25　响应曲线参数设置

域参数包括仿真频率的起止值和步长,以及增益、相角和频率的单位。读者可以根据不同的需要作出选择。

打开 LTI Viewer 仿真环境下的"Tools"菜单,选择"Linestyle Preferences"选项,可以设置响应曲线的各种参数:

图 7-25 上半部分是设置不同的系统和输入输出通道以何种方式区分,有颜色、标志物、线型和不加区分四种选项。下半部分是设置颜色、标志物和线型的优先级。对于大型系统仿真来说,如果希望曲线绘制的更加精美,仿真结果的格式更加规范,就需要在此对话框中选择合适的格式。

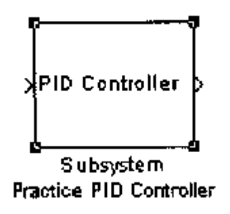
小结:本例详细介绍了 LTI Viewer 环境下一个应用状态反馈进行极点配置的实例。因为是可视化操作,所以图形比较多,显得比较复杂。但事实上使用习惯之后应用 LTI Viewer 进行线性定常系统仿真是非常快捷方便的。另外,在使用 LTI Viewer 过程中不应该忘了 MATLAB 环境下的各种函数,把 MATLAB 强大的数值计算功能和 LTI Viewer 方便的绘图功能结合起来,可以更高效率地完成工作。

7.2 SIMULINK 模块设计

在 SIMULINK 环境下构造大型控制系统方框图时,如果把系统中的每个元件都显示出来,就会遇到空间不够的问题,往往整整一屏都放不下系统方框图的某个局部。解决这个问题的办法就是将实现某一功能的各个元件组合封装起来,形成一个子系统(Subsystem)。整个子系统只占用一个元件大小的位置,可以有效节省空间。并且,将相关的元件组成一个子系统,还可以使系统的结构更加清晰。从外特性上来说,封装后的子系统和普通的元件没有什么太大的区别,同样可以设置参数,通过输入端和输出端与其他的元件相连接。此外,还可以设置子系统的名称和图标,给出子系统的描述和帮助信息。事实上,封装子系统就相当于建立自己的元件,完成特定的功能。

[例 7.3] 考虑第六章构造的实际 PID 控制器的非线性模型如下,试构造该模块的子系统,要求能够设置 PID 和饱和非线性参数,并给出帮助信息。

结果:封装完毕的实际 PID 控制器子系统为:



分析:从图 7-26 实际 PID 控制器的原始模型来看,封装后的子系统应该有一个输入端,一个输出端,6 个参数。这 6 个参数分别是:比例控制器(标有"Proportional"的 Gain 元件)增益 P,积分控制器(标有"Integral"的 Gain 元件)增益 I,积分控制器输出上限 U 和下限 L(标有"Saturation"的饱和非线性元件),微分控制器传递函数的分母多项式系数 D 和分子多项式系数 N(标有"Derivative"的 Transfer Fcn 元件)。因此,构造完的子系统应该能够设置这些参数。

求解过程:

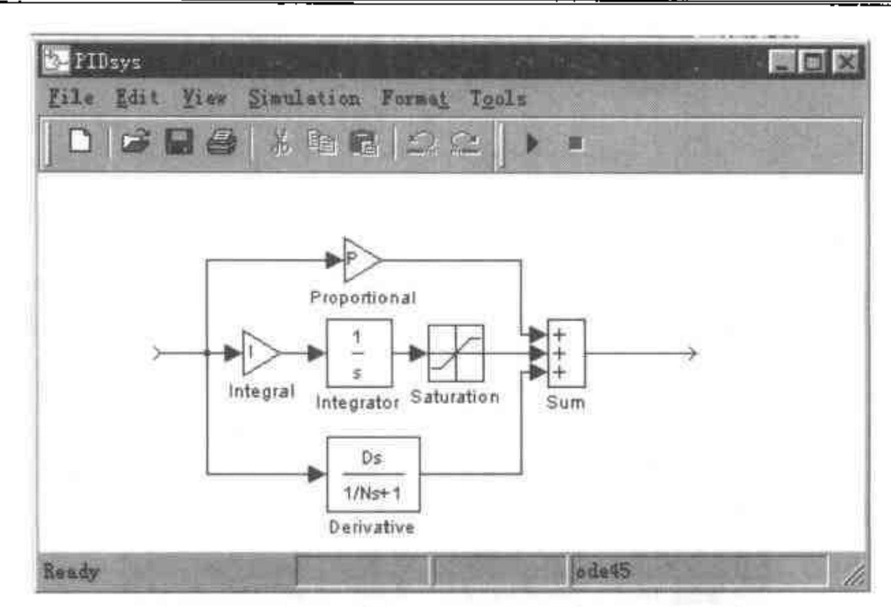


图 7-26 实际 PID 控制器模型

本例的求解分为以下几步：

1. 选择相关元件创建子系统

搭建完系统方框图后首先用鼠标左键选中图 7-26 所示的所有元件，包括连线。然后打开“Edit”菜单，选择“Create Subsystem”选项，如图 7-27 所示。

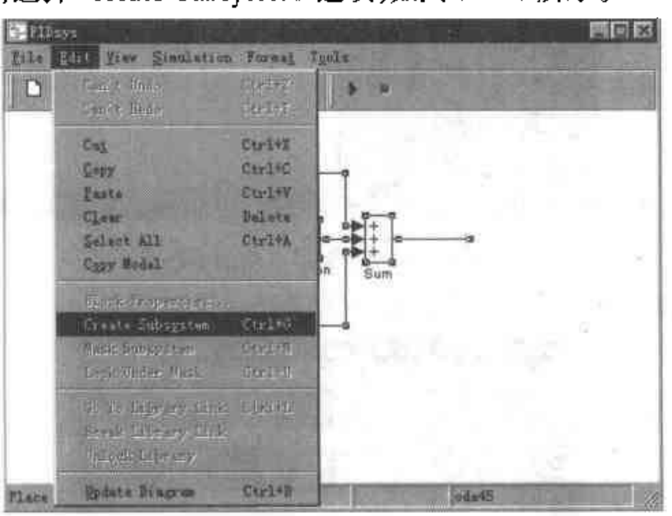


图 7-27 创建子系统

单击该选项后整个系统变为 [Subsystem] 形式的单一元件，输入端为“in1”，输出端为“out1”，标记为“Subsystem”。双击该元件，可还原模型：

从图 7-28 中可以看出，创建完毕的子系统还原后有一个输出端“in1”和一个输入端“out1”，这是因为在原始系统方框图 7-26 中各有一条引入连线和引出连线没有连接元件。如果希望创建完毕的子系统是多输入多输出的，只需在相应的位置引入或引出连线，不连接任何元件即可。

2. 设置子系统参数

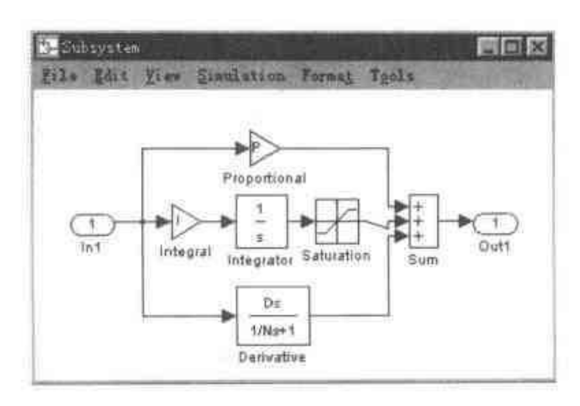


图 7－28 创建子系统的原始模型

创建子系统完毕后，下一步就是具体设置子系统的各个参数打开“Edit”菜单，选择“Edit Mask”选项，如图 7－29 所示。

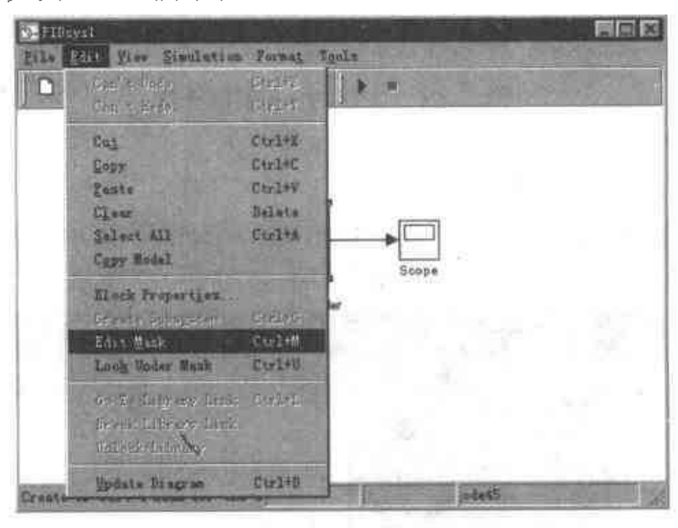


图 7－29 设置子系统的参数

单击该选项即可设置子系统的参数，首先是图标设置界面：

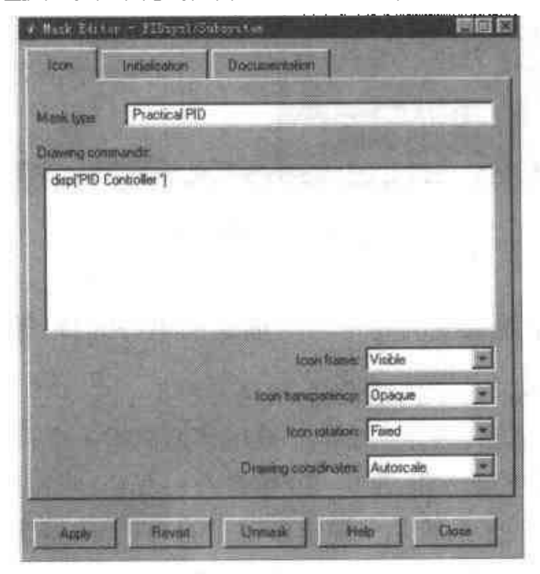


图 7－30 图标设置界面

这里所说的图标就是封装后子系统元件的图标，其功能是提示使用者本元件的作用。图标的设置分为三部分。最上边“Mask Type”文本框是设置子系统的类型，本例填写的是“Practice PID”，表明该子系统是实际 PID 控制器。中间“Drawing Commands”文本框是具体设置图标的命令行，可以用 disp()、plot()、fprintf()等语句设置图标的具体内容。例如本例填写的是：

```
disp('PID Controller');
```

其功能就是显示前边“结果”中设计完毕的子系统图标框内的文字“PID Controller”。该文本框还支持绘图函数 plot()，可以通过简单的曲线表示子系统的功能。事实上，大部分 SIMULINK 自带的元件都是用这种方法设置图标的。例如，我们在该文本框中键入：

```
plot([0 1 5], [0 0 4])
```

该元件的图标就变为如下的形式，在图标框中绘制出一条直线。

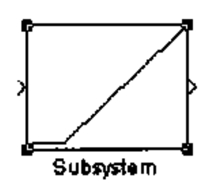


图 7－30 下边的部分是设置图标的一些具体属性，如是否显示边框（Icon Frame）、图标是否透明（Icon Transparency）、是否翻转（Icon Rotation）和坐标格式（Drawing Coordinates）等等。这些选项的功能是改变子系统图标的外观，具体的细节可以参看 SIMULINK 帮助，一般选用缺省设置即可。

点击该界面上方的“Initialization”标签，进入子系统使用变量初始化界面：

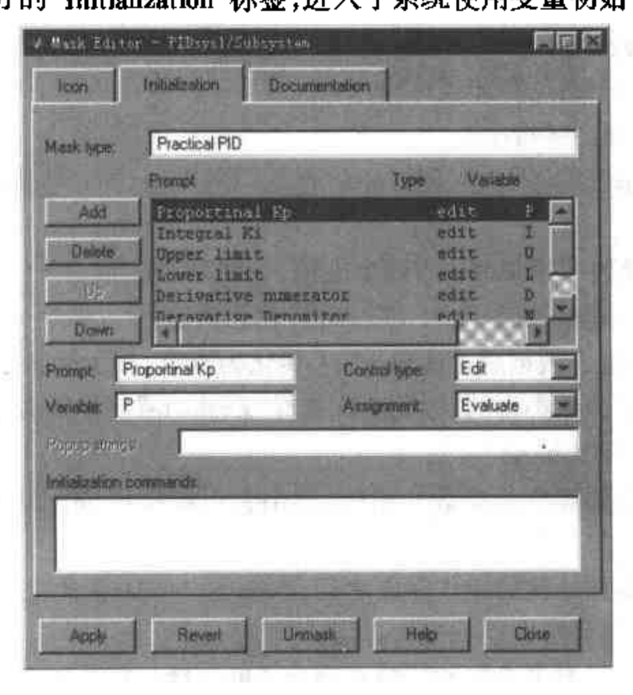


图 7－31　子系统初始化界面

图 7－31 所示界面是子系统设置的主界面，具体的变量参数设置和初始化都是在此界面下完成。

最上边"Mask Type"文本框显示的是子系统的类型，我们在"Icon"界面中曾经设置为"Practice PID"，因此，这里显示的也是"Practice PID"。

中间是子系统变量设置对话框，其各选项的意义见表 7－1。

表 7－1 子系统参数设置功能表

选项名	功 能	选项名	功 能
Add	增加变量	Prompt	变量提示
Delete	删除变量	Control Type	变量类型
Up	变量上移	Variable	对应系统变量
Down	变量下移	Assignment	输入方式

如前所述，本子系统共有 6 个变量需要设置，详见表 7－2。

表 7－2 子系统参数

参数值 / 变量名	Prompt	Control Type	Variable	Assignment
Kp	Proportional Kp	Edit	P	Evaluate
Ki	Integral Ki	Edit	I	Evaluate
D	Derivative Numerator	Edit	D	Evaluate
N	Derivative Denomitor	Edit	N	Evaluate
U	Upper limit	Edit	U	Evaluate
L	Lower limit	Edit	L	Evaluate

其中"Control Type"参数的下拉菜单中有三个选项，分别是：

(1) Edit——通过用户输入；

(2) Checkbox——通过复选框选择；

(3) Popup——弹出字符串。

因为根据实际情况的不同，本例的变量有不同的数值，所以全部选择"Edit"类型，由用户自己输入。

"Assignment"参数下拉菜单中有两个选项，分别是

(1) Evaluate——输入数据后 MATLAB 转化为数值；

(2) Literal——输入数据后 MATLAB 不转化为数值，按字符串处理。

本例的变量都是数值类型的，因此全部选择"Evaluate"。如果需要输入的变量有字符串类型的，就需要选择"Literal"。

图 7－31 最下边的部分是参数初始化，使用者可以按照 MATLAB 规则给前边定义的变量赋值。

有关子系统变量初始化的工作完成之后，点击图 7－31 界面上方的"Documentation"标签，进入子系统文档和帮助信息设置界面：

该界面的功能是设置子系统的说明文档和帮助信息。对于用户自己设计的子系统来说，对系统的功能和变量设置作一些相关说明是十分必要的。图 7－32 界面上边"Block description"是元件的说明，下边"Block help"是元件的帮助信息。填入相关文本后单击元件的"Help"按钮，就可以看到 html 格式或 pdf 格式的用户填写的信息：

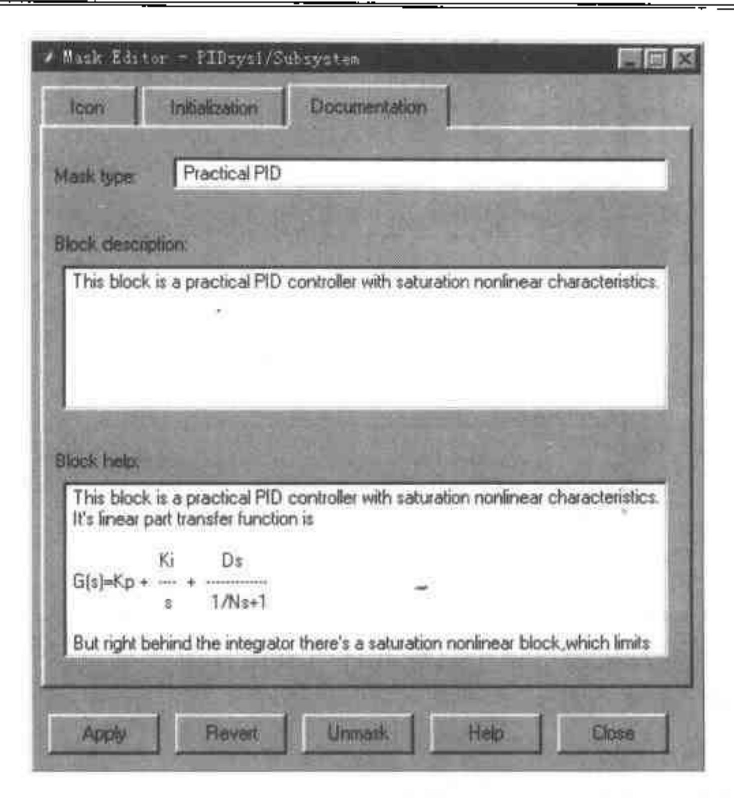


图 7-32 子系统文档设置界面

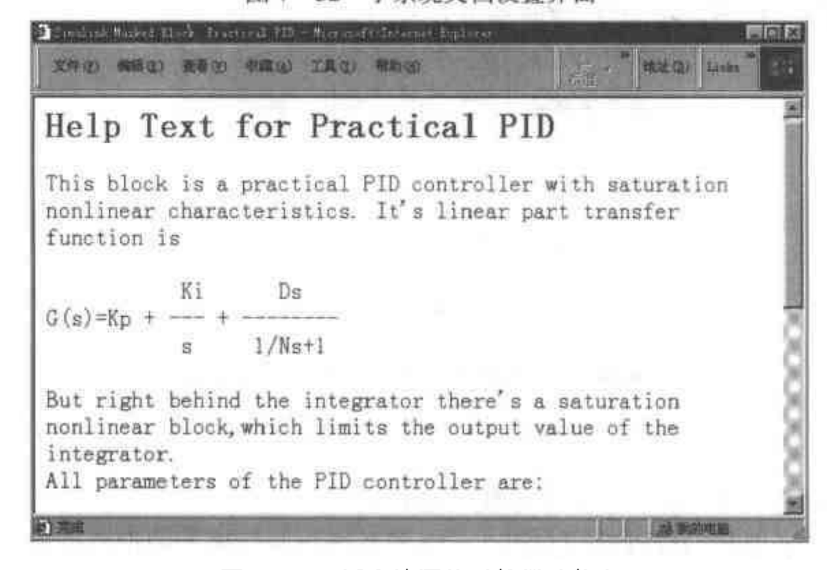


图 7-33 用户填写的元件帮助信息

图 7-33 是单击“Help”按钮后看到的 IE 浏览器下的 html 格式的帮助文件，其内容与图 7-32 子系统文档设置界面中“Block help”文本框的文字是完全一致的。

到此为止，该子系统的主要的参数设置就基本完成了。最后，我们再设置一些元件的外围属性。打开 SIMULINK 仿真环境下的“Edit”菜单，选择“Block Properties”选项，进入元件属性设置界面如图 7-34 所示。

其中“Description”文本框是元件的描述，“Priority”文本框是元件的优先级，“Open Function”文本框是打开元件的函数名称，没有特殊要求的话这些参数都可以不设置。

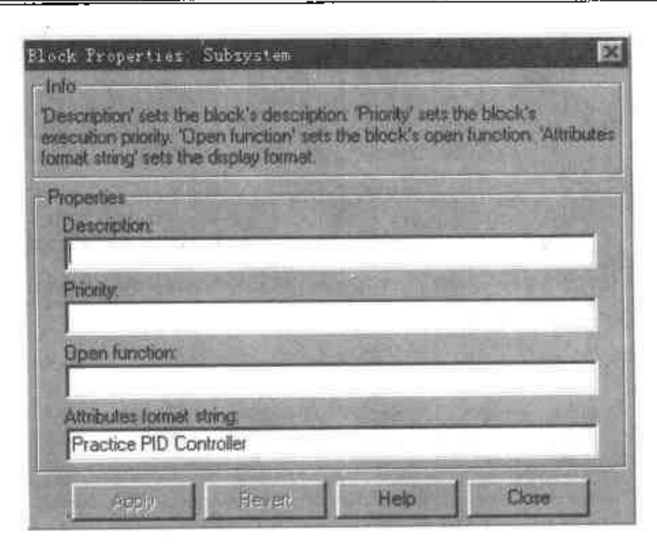


图 7－34 元件属性设置

“Attributes format string”文本框中填写的是显示元件名称的字符串，我们填入“Practice PID Controller”，这样每次调出该子系统元件时其图标下方都显示“Practice PID Controller”，便于用户了解其功能。

3. 子系统仿真

子系统模型建立完成后就可以开始应用仿真了。为了仿真方便我们又调入了一个“Step”元件和一个“Scope”元件，此时系统方框图如图 7－35 所示。

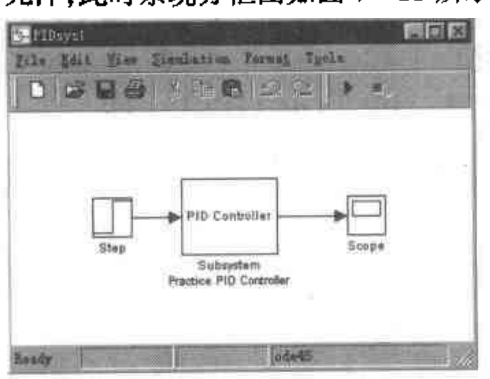


图 7－35 子系统仿真模型

双击图中的“PID Controller”子系统，设置其参数：

从图 7－36 中我们可以看到在图 7－32 子系统初始化界面中设置的各个参数以及图 7－33设置的说明信息，此时点击“Help”按钮还可以看到上一步设置的帮助信息。按照图中的数值填写完毕后即可开始仿真，仿真曲线为：

从图 7－37 中可以看到，系统的阶跃响应曲线和第六章里没有构成子系统的实际 PID 控制器响应曲线相同，说明构建的子系统可以正常工作。

小结：本例介绍了 SIMULINK 环境下构建子系统的方法，从实际应用的效果来看，一方面简化了系统的方框图，另一方面也增加了系统的可靠性，同时还为以后的工作做出了铺垫。这里尤其需要注意的是，应该认真填写帮助文档，否则有可能对以后的工作造成不必要的麻烦。

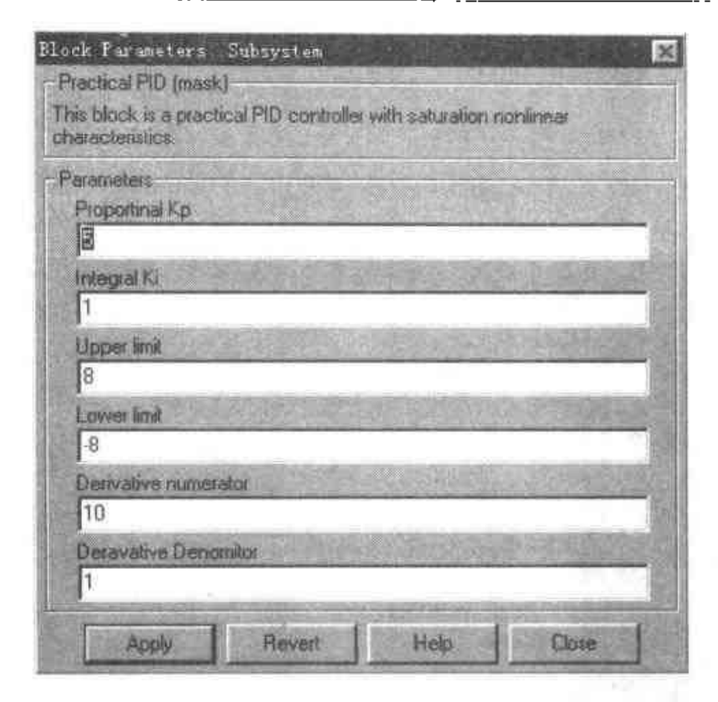


图 7-36 PID Controller 子系统仿真参数设置

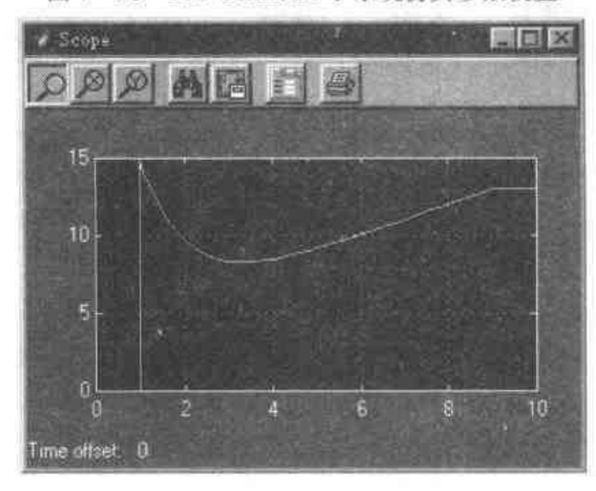


图 7-37 PID 子系统仿真曲线

7.3 SIMULINK 仿真举例

在第六章我们曾经提到,SIMULINK 中内置了许多例子,读者可以通过这些例子来学习 SIMULINK 环境的使用和标准的系统建模、仿真方法。不过这些例子一般都比较复杂,不太适合初学者使用,因此在前边的讲解中我们没有采用这些 SIMULINK 中内置的例子,而是通过一些更基本的、与控制系统联系更紧密的简单实例使读者逐渐熟悉 SIMULINK 环境及其功能。

事实上,有了前边的基础之后,读者基本上就可以大致掌握并自己学习 SIMULINK 中内置的例子了。

7.3.1 MATLAB 浏览程序

除了第六章里介绍的进入 SIMULINK 仿真环境后选择"Demo"图标查看 SIMULINK 中内置的例子之外，还可以在 MATLAB Command Window 下直接键入"tour"，进入如图 7-38 所示界面。

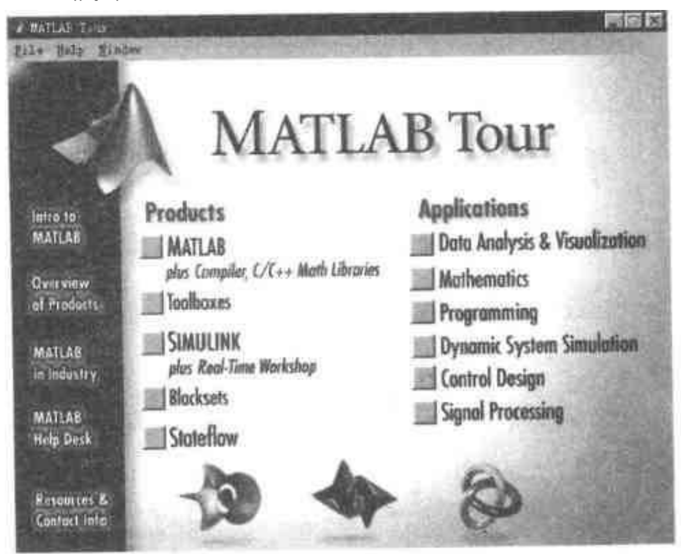


图 7-38 MATLAB 浏览界面

该界面包含了 MATLAB 环境下各种基本语法和仿真工具的介绍以及内置的例子。在具体讲述 SIMULINK 的例子之前，我们先介绍一下该 MATLAB 浏览界面的主要内容，方便读者的查询。

1."MATLAB, plus Compiler, C/C++ Math Libraries"。该选项包括 MATLAB 环境的基本介绍、帮助、运行方式、语法规则，本书第一章介绍的主要内容基本都可以在这里找到。此外还有 MATLAB 编译器、MATLAB C/C++数学库的一些信息，希望在 C/C++环境中调用 MATLAB 函数的读者应详细阅读这部分内容。

2."Toolbox"。该选项介绍了 MATLAB 内置的各种工具箱，包括控制系统工具箱(Control System Toolbox)、鲁棒控制工具箱(Robust Control Toolbox)、模型预测控制工具箱(Model Predictive Control Toolbox)、mu 分析与校正工具箱(mu Analysis and Systhesis Toolbox)、多变量系统频域设计工具箱(Multi-Variable Frequency Design Toolbox)、定量反馈控制工具箱(Quantitative Feedback Theory Toolbox)、频域系统辨识工具箱(Frequency Domain System Identification Toolbox)、系统辨识工具箱(System Identification Toolbox)、神经网工具箱(Neural Network Toolbox)、小波分析工具箱(Wavelet Toolbox)、最优化工具箱(Optimization Toolbox)、偏微分方程工具箱(Partial Differential Equation Toolbox)、信号处理工具箱(Signal Processing Toolbox)、图像处理工具箱(Image Processing Toolbox)、扩展符号数学工具箱(Extended Symbolic Math Toolbox)。该选项包括各个工具箱的功能、实例和帮助，具体各工具箱的函数请参阅本书的目录。

3."SIMULINK plus Real-Time Workshop"。该选项介绍 SIMULINK 仿真环境，下边介绍的实例就在该选项中。其主要内容有：

(1)"New in Simulink 2"。介绍 SIMULINK 2 的一些新的特性以及帮助，适合第一次接触 SIMULINK 环境或 SIMULINK 2 的读者使用。

(2)"Simple Models"。SIMULINK 的一些简单例子，其难度和本书第六、七章介绍的实例大体相当，但主要是演示功能，适合有一定基础的读者。不过想要系统学习，还是应该参考相应的教材。

(3)"Complex Models"。SIMULINK 的一些复杂例子。其难度接近大型工程实际应用的水平，包括著名的 F-14 战机线性化仿真模型的例子，适合有一定实际经验的读者参考。我们下文举的例子就是从这个选项中选取的。

(4)"Advanced Products"。该选项主要是 Real-Time Workshop 介绍，讲述根据 SIMULINK 模型生成 C 源代码的方法。

4."Blochsets"。该选项介绍 MATLAB 封装的各种模块组，类似于 SIMULINK 中的 Library。包括通信模块(Communication)、离散信号处理模块(DSP Blocksets)、电力系统模块(Power System Blocksets)等等。

5."Stateflow"。该选项介绍了一种新的 SIMULINK 环境下的绘图及仿真环境——状态流。该仿真环境能够用于复杂控制及逻辑状态监控，使 SIMULINK 可以将连续仿真和事件驱动仿真两种方式结合起来，在一个闭环系统之下实现。该仿真环境有较高的针对性和实用性，适合专业设计人员使用。

6."Data Analysis & Visualization"。该选项介绍了数据统计分析和 MATLAB 环境下的各种二维、三维绘图以及可视化工具。本书第一章的非线性拟合以及绘图的内容都在这里有所介绍。

7."Mathematics"。MATLAB 数学库函数。

8."Programming"。MATLAB 语法规则及编程技巧。

9."Dynamic System Simulation"。该选项介绍了 MATLAB 环境下的各种动态系统仿真工具，包括 SIMULINK、Stateflow、DSP Blocksets、Fuzzy Logic 等等，包括基本功能及相关实例。

10."Control Design"。该选项囊括了各种与控制系统相关的设计和仿真环境、工具箱，包括：控制系统工具箱(Control System Toolbox)、SIMULINK、系统辨识工具箱(System Identification Toolbox)、定量反馈工具箱(QFT Control Design Toolbox)、神经网络工具箱(Neural network toolbox)、鲁棒控制工具箱(Robust Control Toolbox)等等。本书第二至五章用到的函数在这里都有介绍。

11."signal Processing"。该选项的内容是各种与信号处理相关的工具箱，包括信号处理工具箱(Signal Processing Toolbox)、小波分析工具箱(Wavelet Toolbox)、图像处理工具箱(Image Processing Toolbox)、最优化工具箱(Optimization Toolbox)、模糊逻辑工具箱(Fuzzy Logic Toolbox)等等。事实上，同是以系统论、控制论和信息论为基础的控制领域和信号处理领域在近几十年的发展中一直是相互影响、相互促进，控制理论中的拉普拉斯变换和频域分析的方法最初都是从信号处理领域中借鉴过来的。因此，掌握一些信号处理的内容对深入学习控制理论是非常必要的。

以上就是 MATLAB 浏览界面的主要内容，点击各个选项前面的按钮就可以进入前面介绍的相应的各分类内容。本节的主要内容是介绍 SIMULINK 内置的例子，因此点

击 SIMULINK 按钮,进入如图 7－39 所示界面。

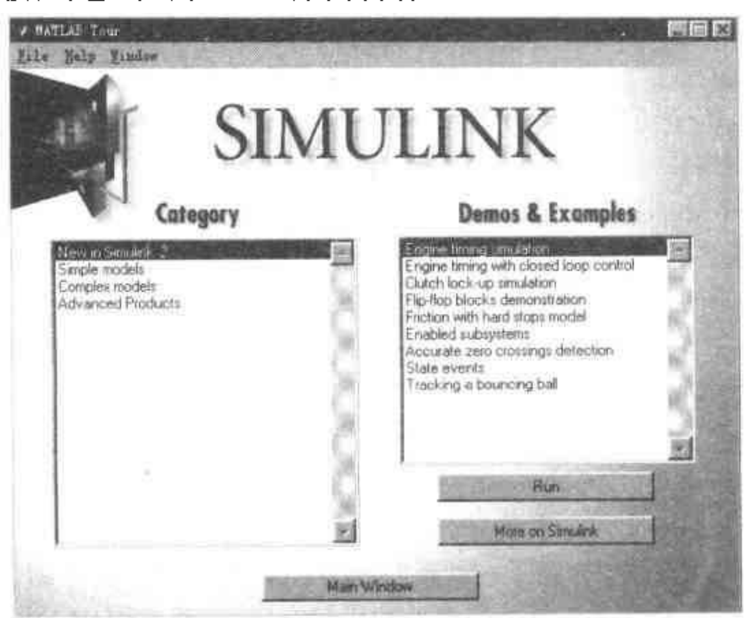


图 7－39　SIMULINK 浏览界面

该界面下我们可以看到前边介绍的 New in Simulink 2、Simple Models、Complex Models、Advanced Products 等内容。下边我们就给出 Complex Models 里的一个实例,提高读者分析和设计控制系统的能力。

7.3.2　仿真举例

选择“MATLAB Tour”界面的 SIMULINK 按钮,进入 SIMULINK 浏览界面后选择“Complex Models”选项,双击“Pendulum with two joints”,即可进入图 7－40 所示的双轴单摆仿真方框图。

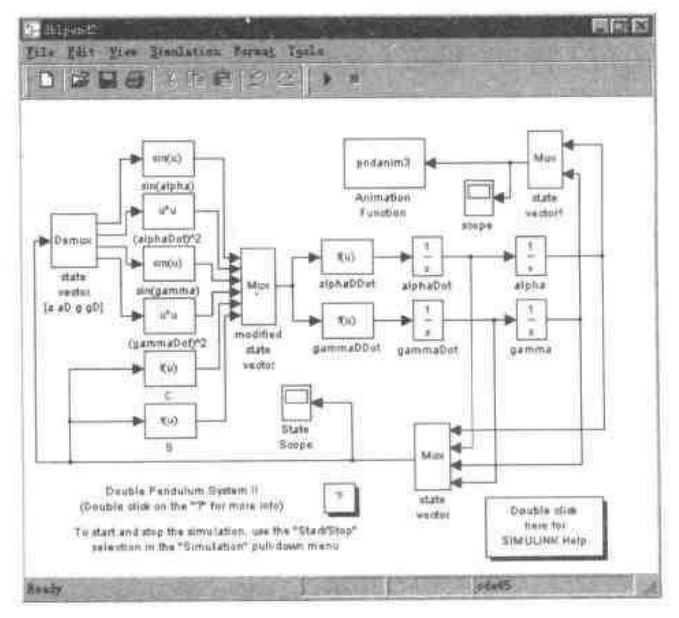


图 7－40　双轴单摆仿真方框图

该仿真方框图中包含了两种我们以前没有接触过的新元件，一种是“Fcn”函数元件，其功能是设定元件的输入输出为任意的函数关系(一般是非线性函数)。双击此元件图标，可以进入其参数设置对话框，如图 7-41 所示。

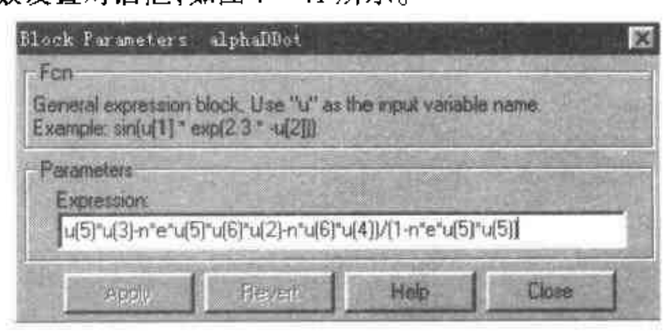


图 7-41 函数元件参数设置对话框

该元件使用“u”作为输入变量的名称(注意，无论输入值的具体名称是什么，一概以 u 代替)，u(1)表示输入向量的第一个分量，其余以此类推。具体的函数表达式与 MATLAB 语法完全一致，只要是合法的 MATLAB 表达式即可。

另外一个元件就是标有“S - Function”的 S 函数元件。尽管 SIMULINK 提供了十分丰富的模型库，但在研究复杂系统时会遇到一些特殊的环节，不能用给定的模型元件来描述。SIMULINK 允许用户自己利用 S 函数创建特殊的模型元件。这大大地扩大了 SIMULINK 的应用领域，这一功能也充分体现了 MATLAB 开放性、可扩充性这一特点。双击此元件图标，可以看到其参数设置对话框，如图 7-42 所示。

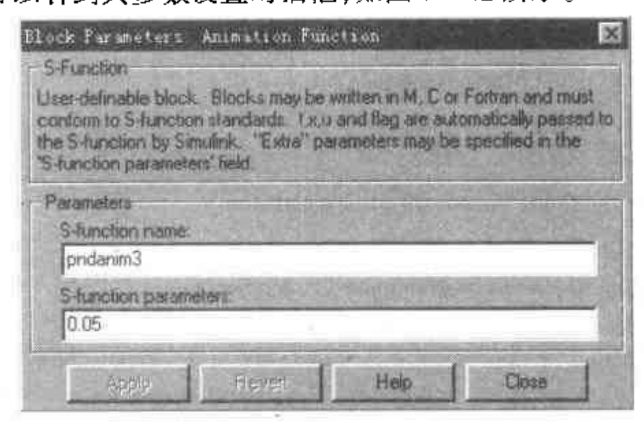


图 7-42 S 函数元件参数设置对话框

S 函数实质上是具有特殊调用格式的 MATLAB 函数，表征系统动态特性，能使 SIMULINK 有能力构造一般的仿真方框图，灵活方便地设计出用户需要的特殊环节。实际上，SIMULINK 的每一个模型都可以用输入、输出、状态三个要素进行描述，S 函数的描述同样遵循这一原则。通常 S 函数的调用格式为：

function[sys, x0] = sfunc(t, x, u, flag, para1, para2, ...)

其中，sfunc 是用户定义的系统，t 是当前时刻，x 是当前状态值，x0 是初始状态值，u 是当前系统输出值，变量 flag 的值控制返回变量 sys 的信息。我们通过图 7-42 的 S 函数元件参数设置对话框可以查到该元件相应的 S 函数代码如下：

```
function [sys,x0] = pndanim1(t,x,u,flag,ts);
```

```
% PNDANIM3 S-function for animating the motion of a pendulum.
%       Ned Gulley, 6-21-93
%       Copyright (c) 1990-1998 by The MathWorks, Inc. All Rights Reserved.
%       $Revision: 5.5 $

global PendAnim1

if flag == 2,
    if any(get(0,'Children') == PendAnim1),
      if strcmp(get(PendAnim1,'Name'),'simppend Animation'),
        set(0,'currentfigure',PendAnim1);
        hndlList = get(gca,'UserData');
        x = [u(1) u(1) + 2 * sin(u(2))];
        y = [0  - 2 * cos(u(2))];
        set(hndlList(1),'XData',x,'YData',y);
        set(hndlList(2),'XData',u(1),'YData',0);
        drawnow;
      end
  end
  sys = [];
elseif flag == 4 % Return next sample hit
   % ns stores the number of samples
   ns = t/ts;
   % This is the time of the next sample hit.
   sys = (1 + floor(ns + 1e-13 * (1 + ns))) * ts;
elseif flag == 0,
   % Initialize the figure for use with this simulation
   animinit('simppend Animation');
   [flag,PendAnim1] = figflag('simppend Animation');
   axis([-3 3 -2 2]);
   hold on;
   x = [0 0];
   y = [0 -2];
   hndlList(1) = plot(x,y,'LineWidth',5,'EraseMode','background');
   hndlList(2) = plot(0,0,'.','MarkerSize',25,'EraseMode','back');
   set(gca,'DataAspectRatio',[1 1 1]);
   set(gca,'UserData',hndlList);
   sys = [0 0 0 2 0 0];
   x0 = [];
end
```

阅读这段代码后发现，这是一段根据系统的输入显示动画的程序，其核心部分是根据不同的 flag 值(即系统的不同状态)进行计算，然后将计算结果绘制成图形，将这些图形连续显示，就构成了动画。其中 animinit()函数、hndlist()函数和 set()函数是仿真动画的核

心语句，感兴趣的读者可以查阅 MATLAB 帮助。

事实上，本例的仿真结果也正是双轴单摆的动画演示。图 7-43 给出了其中的一幅仿真动画示意图，此时单摆与垂直方向大约成 30°夹角，而读者在 MATLAB 下仿真时就会看到此单摆围绕两个轴不停的摆动。

在 SIMULINK 环境下，用户可以方便地进行控制系统建模和仿真，这一点相信通过阅读本书的第六、七两章读者已经有所了解了。事实上，MATLAB 5.3 版本下的 SIMULINK 3.0 还有许多扩展的功能，例如设置模块的优先权、合并输入信号、以及更新的数据结构等等。通过阅读 MATLAB 帮助读者对其有所了解。

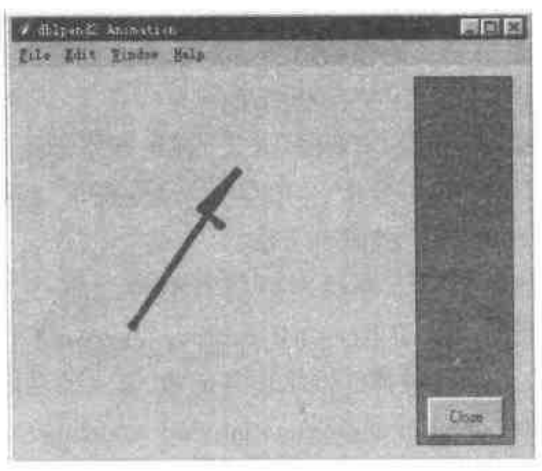


图 7-43 双轴单摆仿真动画示意图

7.4 小 结

本章我们主要介绍了一些控制系统设计和仿真领域常用的 SIMULINK 高级技术，包括线性定常系统仿真工具 LTI Viewer 的环境设置与使用方法、SIMULINK 环境下的模块设计技术、以及 SIMULINK 环境下复杂控制系统仿真举例。其中线性定常系统仿真工具 LTI Viewer 是一个非常不错的线性系统仿真环境，尤其适合控制领域的初学者使用。SIMULINK 环境下的模块设计技术给我们提供了一种面向问题的拓展手段，为解决特殊要求控制系统仿真的问题打下了基础。

关于 SIMULINK 仿真的内容我们就介绍到这里，下一章我们将介绍有关 MATLAB 环境下神经网络工具箱的内容。

第八章 神经网络与控制

神经网络是 20 世纪 80 年代末期发展起来的一种实用的多学科交叉技术,已成为“智能控制”的一个新分支,是自动控制领域的前沿学科之一。它为解决复杂的非线性、不确定、不确知系统的控制问题,开辟了一条新的途径。

本章介绍的神经网络控制理论主要是(人工)神经网络理论与控制理论的结合,而神经网络技术发展至今,已经汇集了数学、生物学、神经生理学、脑科学、遗传学、人工智能、计算机科学等多领域学科的理论、技术和方法。

MATLAB 环境下提供的用于神经网络仿真的神经网络工具箱(Neural Networks Toolbox),目前已经发展到 Neural Network Toolbox Version 3.0.1 版本了。该版本提供了功能更强大、界面更友好的各种神经网络仿真函数,其主要功能包括:

1. 函数逼近与建模(Function approximation and modeling)。
2. 信号处理与预测(Signal processing and prediction)。
3. 分类与聚类(classification and clustering)。

读者可以在 MATLAB Command Window 下键入“demo”命令,通过图 8－1 的界面了解神经网络工具箱新增的内容。

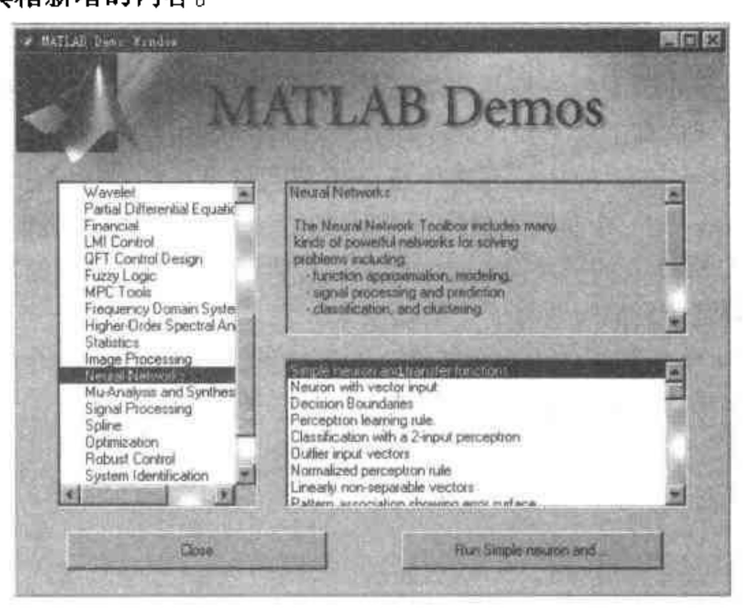


图 8－1 MATLAB 神经网络工具箱演示界面

本章首先介绍一些关于神经网络控制的基础知识,然后结合 MATLAB 环境具体阐述神经网络工具箱的使用方法,最后给出一个应用的实例。

8.1 神经网络概述

人工神经网络(简称神经网络,Neural Networks)是人工神经元(简称神经元,Neuron)相互连接而组成的网络,它是从微观结构和功能上对人脑的抽象、简化,是模拟人类智能的一条重要途径,反映了人脑功能的若干基本特征,如并行信息处理、学习、联想、模式分类、记忆等等。

1943年建立的第一个神经元模型——MP模型,为神经网络的研究与发展奠定了基础。至今,已经建立了多种神经元与网络的模型,取得了相当多的成果。其中神经网络对控制领域最有吸引力的特性是:

1. 能任意逼近L2上的非线性函数;
2. 可以实现同步多输入、多输出;
3. 便于用超大规模集成电路(VLSI)或光学集成电路系统实现,或用现有的计算机技术实现;
4. 能够实现信息的并行分布式处理与存储;
5. 能够进行自学习、自调整,适应环境的变化。

图8-2是控制领域常用的多层前馈神经网络模型:

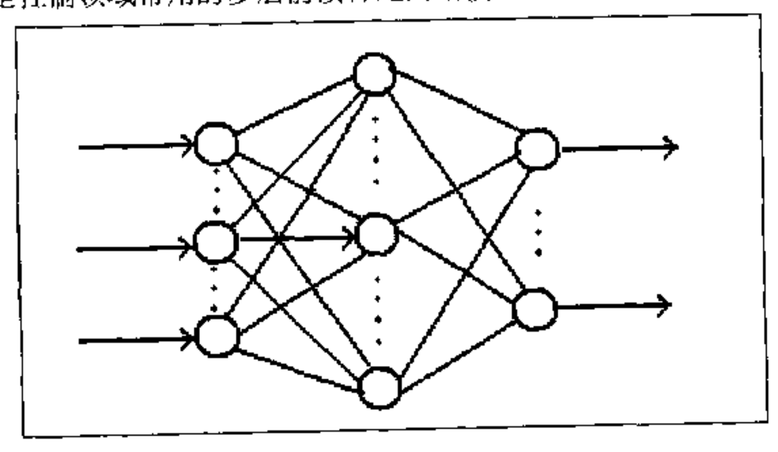


图8-2 多层前馈神经网络模型

8.1.1 神经网络理论基础

人工神经网络或称连接机制(Connectionism)是源于人脑神经系统的一类模型,它是由简单信息处理单元(人工神经元)相互连接组成的网络。网络的信息处理又通过处理单元之间的相互作用来实现,具体地说,它是通过把问题表达成处理单元之间的连接权值来处理的。权值大,则该处理单元对后续单元的影响大,反之亦然。

自20世纪80年代末期以来,学者们建立了多种神经网络模型,其理论框架和思路不外乎下述三大要素:

1. 神经元(信息处理单元)的特性。
2. 神经元之间的相互见解形式——网络拓扑结构。

3．为适应环境而改善性能的学习规则。

神经网络是具有高度非线性的系统，具有一般非线性系统的特性。虽然单个神经元的组成和功能极其有限（一般只是权值调整），但大量神经元组成的网络系统，所能实现的功能确是非常丰富多彩的。

神经网络的数学模型虽然有很多种，但基本运算可以归结为四类：积与和、权值学习、阈值处理和非线性函数处理。其中关键的部分是权值的学习机制，它是模拟人适应环境的学习过程的一种机器学习模型，大致分为以下三种：

1．有导师学习（SL——Supervised Learning）　在学习过程中，网络根据实际输出与期望输出的比较，进行连接权值的调整。一般将期望输出称为导师信号，它是评价网络学习的标准。

2．无导师学习（NSL——Nonsupervised Learning）。在学习过程中，没有导师信号提供给网络，网络能根据其特有的结构和学习规则，进行连接权值的调整。此时，网络的学习评价标准隐含于其内部。

3．再励学习（RL——Reinforcement Learning）。它把学习看作为试探评价（奖励或惩罚）过程，学习机制选择一个动作（输出）作用于环境后，使环境的状态改变，并产生一个再励信号 r_e（奖励或惩罚）反馈至模型，模型依据再励信号与当前环境的状态，再选择下一个动作作用于环境。动作选择的原则，是使受到奖励的可能性增大。

下面我们以最简单的 MP 模型为例，具体阐述神经网络的工作机制。

MP 模型是 1943 年由美国心理学家 McCulloch 和数学家 Pitts 共同建立的，是最早的人工神经元模型。它是一个多输入——多输出的非线性信息处理单元，其具体的网络拓扑结构如图 8－3 所示。

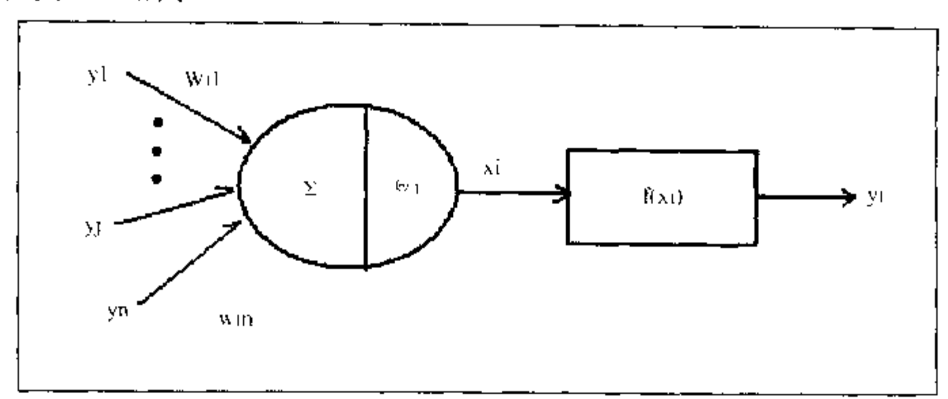


图 8－3　MP 人工神经元模型结构

其中，y_i——神经元 i 的输出，它可以与其他多个神经元通过权相连接；

y_j——与神经元 i 连接的神经元 j 的输出，也是神经元 i 的输入；

w_{ij}——神经元 i 到神经元 j 的连接权值；

θ_i——神经元 i 的阈值；

$f_{(xi)}$——输出非线性函数。

神经元 i 的输出 y_i 可以用下式描述：

$$y_i = f\left(\sum_{j=1}^{n} W_{ij} y_j - \theta_i\right),\quad i \neq j$$

设：
$$x_i = \sum_{j=1}^{n} W_{ij} y_j - \theta_i$$
则有：
$$y_i = f(x_i)$$

每一个神经元的输出，或者是数值“0”，或者是数值“1”，分别模拟生物神经的“抑制”或“兴奋”状态。则，输出非线性函数为：

$$f(x) = \begin{cases} 1, & x \geq 0 \\ 0, & x < 0 \end{cases}$$

f(x)是一个作用函数(Activation Function)，也称激发函数。上式是最简单的阶跃函数形式的作用函数，常用的还有 Sigmoid 型作用函数：

$$f(x) = \frac{1}{1 + e^{-x}}$$

这两种作用函数的图形如下所示：

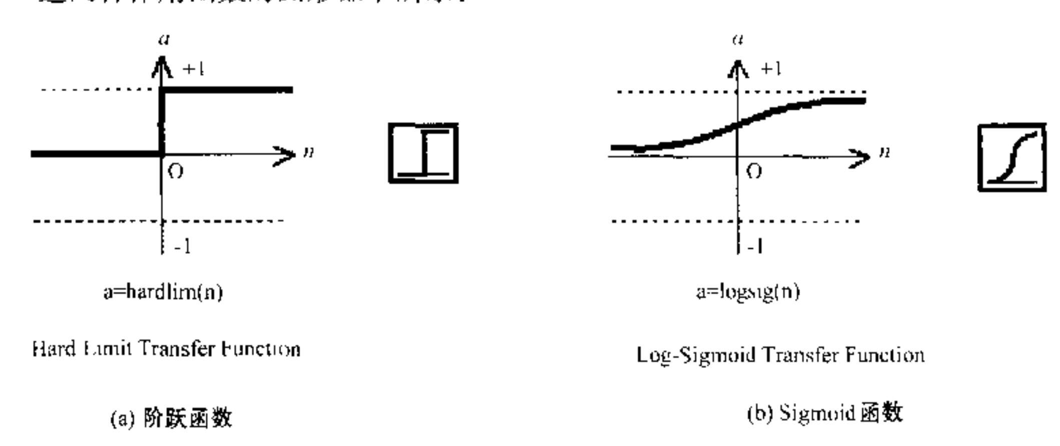


图 8-4　常用的作用函数类型

有时为了满足对称性的要求，也采用下述形式的对称 Sigmoid 型作用函数：

$$f(x) = \frac{1 - e^{-\beta x}}{1 + e^{-\beta x}}, \quad \beta < 0$$

其图像就是将图 8-4(b)向下平移 0.5 个坐标单位，并伸长 2 倍，使其关于 x 轴对称的同时仍保持上限为 1。同样，阶跃函数也可以做上述对称变换，得到对称型阶跃函数：输入小于零时其输出为 -1，输入大于零时其输出为 1。

MP 模型是最简单的人工神经网络模型，但作为一种理论雏形来说已经足够了。剩下的问题就是如何改善网络拓扑结构和寻求权值学习算法。例如目前流行的多层前馈网络和 BP 算法，就是对网络拓扑结构和权值学习算法的修正。相关的内容我们将在后边结合实例进行具体介绍。

8.1.2　神经网络控制

神经网络用于控制领域，主要是为了解决复杂的非线性、不确定、不确知系统的控制问题。由于神经网络具有模拟人的部分智能的特性，主要是具有学习能力和自适应性，使神经网络控制能对变化的环境具有自适应性，并且成为基本上不依赖于模型的一类控制。因此，神经网络控制已成为“智能控制”的一个新的分支。

到目前为止,神经网络在包括 PID 控制在内的控制领域各种模型中都有所应用,其核心算法就是利用神经网络的高度非线性来逼近各种难控模型,再通过一定的校正手段达到控制系统的设计性能指标。

目前常用的有以下几种神经网络控制方式:

1.PID 控制。PID 控制是最常用的一种控制方法,也是我们曾经详细分析过的。利用神经网络进行 PID 控制,通过神经控制器(NNC)和神经网络辨识器(NNI)进行参数调整,能够起到智能控制的作用。其结构见图 8-5 所示。

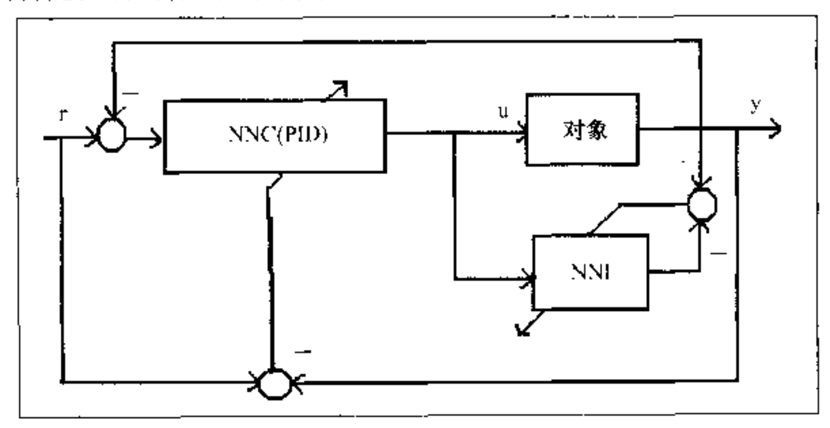


图 8-5 神经网络 PID 控制

2.直接逆动态控制。利用 NNC 与被控对象串联,NNC 实现对象 P 的逆模型 P^{-1}。其优点是能够在线调整模型参数,实现设定值跟踪。

3.间接自校正控制。由 NNI 对被控对象进行在线辨识,根据"确定性等价"原则,设计控制器参数,以达到有效控制的目的。

4.模型参考自适应控制。利用 NNC 和 NNI 构造对象的参考模型,根据神经网络的自调整功能实现在线辨识和控制。

5.内模控制。神经网络内部模型控制(IMC——Internal Model Control),就是首先利用 NNI 对被控对象 P 进行在线辨识,然后用 NNC 实现对象 P 的逆模型 P^{-1},再加入滤波器提高系统鲁棒性的一种控制方式。其控制器输生由被控对象与内部模型的输出误差来调整。

6.前馈反馈控制。利用 NNC 构造前馈控制器,常规控制器来构造反馈控制器,可以有效的抑制扰动的作用。

7.预测控制。预测控制是一种基于模型的控制,其算法三要素是模型预测、滚动优化和反馈校正。利用神经网络良好的非线性函数逼近性能,可以实现非线性对象的预测模型,从而保证优化目标的实现。

以上神经网络技术在控制领域的一些应用,事实上神经网络的功能远不止如此,下面我们结合 MATLAB 神经网络工具箱具体介绍神经网络的一些用途。

8.2 MATLAB 神经网络工具箱

如前所述,目前 MATLAB 神经网络工具箱最新的版本是 Version 3.0.1。我们从前

边介绍过的最简单的 MP 模型入手,引出非线性函数逼近,并逐步介绍其功能。

8.2.1 MP 模型仿真

MP 模型是神经网络中最简单的单元,也是 MATLAB 神经网络工具箱的最基本功能。请看下面这个例子。

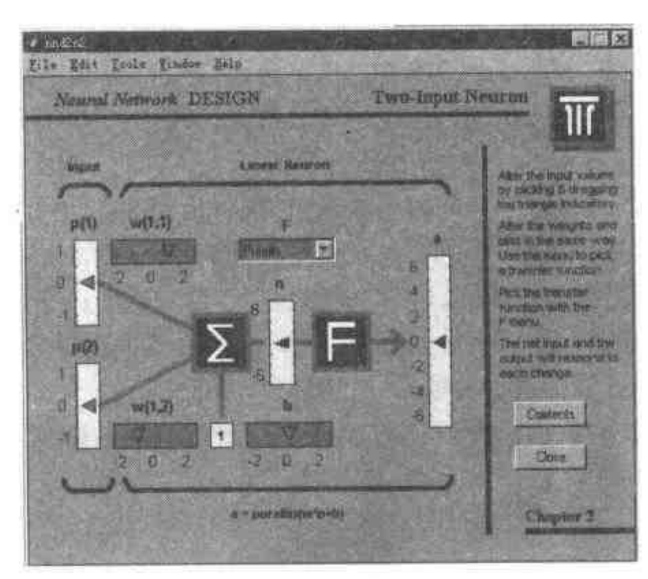


图 8-6 简单神经元 MP 模型

[例 8.1] 在图 8-1MATLAB 神经网络工具箱演示界面下选择"Simple neuron and transfer functions"选项,即可进入如图 8-6 所示的简单神经元 MP 模型演示界面。不过最初进入的演示界面只有一个输入量 p,可以点击图 8-6 界面右下方的第一个按钮"contents",进入简单神经元模型内容设置界面如图 8-7 所示。

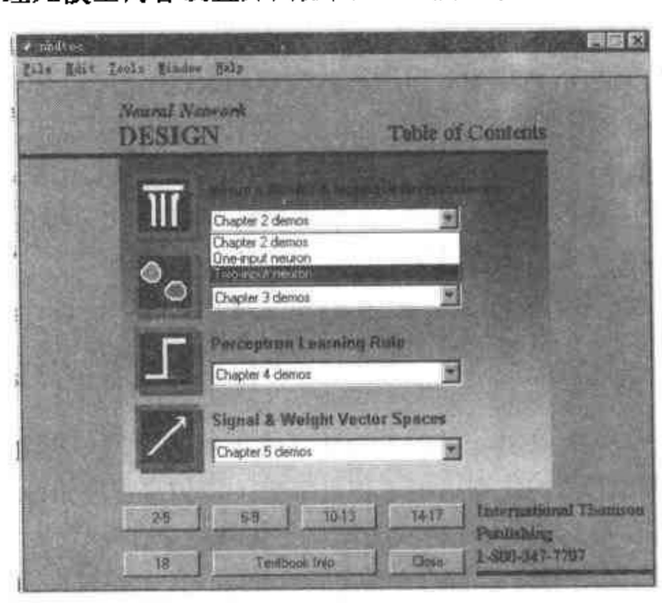


图 8-7 简单神经元模型内容设置界面

该界面包括如下四个选项:

1. 神经元模型与网络体系结构(Neuron Model & Network Architectures)。
2. 演示实例(An Illustrative Example)。
3. 感知器学习规则(Perceptron Learning Rule)。
4. 信号与权重向量空间(Signal & Weight Vector Spaces)。

我们只需在第一个选项的下拉菜单中选择"Two Input Neuron"选项即可,其余选项均使用 MATLAB 的缺省设置。

下面我们回到图 8-6 所示的 MATLAB 神经网络工具箱演示界面。该界面中间的简单神经元模型与图 8-3 所示的神经网络 MP 模型是完全一致的,只是各个变量字母代表的具体含义不同。

其中 p(1)和 p(2)代表神经元的两个输入量,也就是图 8-3 中的 y_i;w(1,1)和 w(1,2)代表两个输入量连接到该神经元的权值,也就是图 8-3 中的 w_{ij};b 是神经元的阈值,也就是图 8-3 中的 θ_i;∑框代表求和,这一点与图 8-3 是一致的;F 框就是图 8-3 中的非线性作用函数,可以通过该界面下中间标有"F"的下拉菜单中选择具体的函数形式,其菜单格式见下页图 8-8。a 则表示神经元的输出。

根据上一节的讨论,我们选择对称的 Sigmoid 型作用函数,即从"F"的下拉菜单中选择"logsig"选项。

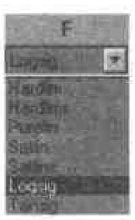


图 8-8　作用函数类型选择

图 8-6 界面的最下方,还给出了该神经元的仿真模型公式:

$$a = \mathrm{logsig}(w \cdot p + b)$$

确定了输入量、网络拓扑结构和作用函数类型后,我们就可以对该简单神经元网络模型进行仿真了。该演示模型的仿真完全是可视化的,只需用鼠标拖动图中相关参数下边黄色或灰色刻度条里边的红色三角即可改变其参数值(注意,输出量 a 刻度条的黑色三角是不能拖动的,因为这是最终计算的值)。最终的计算结果由 F 框后面的红色箭头显示出来。请看如图 8-9 所示的计算结果:

在图 8-9 中,我们令两个输入量 p(1) = 1, p(2) = -1;两个权值 w(1,1) = 1, w(1,2) = 1;阈值 b = 1。从 F 框后面的红色箭头指示的结果来看,该简单神经元的输出结果为0.8左右。

小结:本例比较详细的叙述了 MATLAB 环境下如何对简单神经元 MP 模型进行仿真的问题,并根据临时选择的数据给出了仿真结果。该仿真程序秉承了 MATLAB 5.x 可视化的风格,用鼠标即可完成所有的操作,简化了仿真的工作量。

第一次接触神经网络的读者可能会提出这样的疑问,上面这些工作不借助神经网络工具箱,在 MATLAB Command Window 下同样也可以完成,神经网络技术具体有什么作用呢?事实上,上面那个例子是最简单的一个神经元的网络,而实际应用的神经网络中就

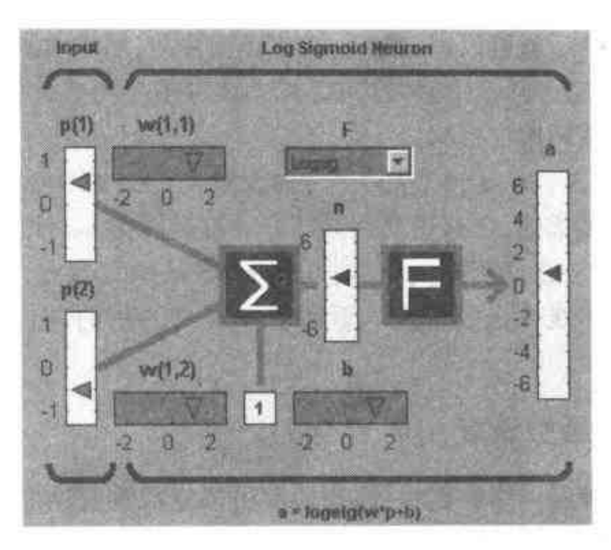


图 8-9　简单神经元计算结果

远大于此了。这么多神经元，没有相应的函数计算，就非常复杂难解了。神经网络正是通过这些庞大的神经元实现了高度非线性，从而能够任意逼近 L_2 空间中的函数。下面我们首先介绍一些 MATLAB 神经网络工具箱中的函数，然后再通过一个非线性函数逼近的例子具体阐述这些函数的用法。

对于给定的由一组输入向量和输出向量描述的非线性函数，可以首先用 errsurf()函数来评价其对于单个神经元的非线性程度。该函数的调用格式为：

e = ERRSURF(P,T,WV,BV,F)

其中 P 是 Q 维输入向量，T 是 Q 维输出向量；WV 是单个神经元的输出权值向量，BV 是相应的阈值向量；F 是字符串形式的作用函数。返回值 e 是相对应于不同权值和阈值的误差矩阵。

得到单个神经元的误差矩阵后可以用 plotes()函数将其以三维图形的方式绘制出来，其调用格式为：

PLOTES(WV,BV,ES,V)

其中 WV 是单个神经元的输出权值向量，BV 是相应的阈值向量；ES 是误差矩阵；V 是图形坐标设置，缺省值为[-37.5, 30]。

在利用神经网络逼近非线性函数时，首先要创建神经网络。常用的是 newlin()函数，其功能是创建一层线性神经网络，调用格式为：

net = NEWLIN(PR,S,ID,LR)

其中 PR 是 R 个输入向量的最小和最大值矩阵(或向量)；S 是输出向量个数；ID 是输入延迟向量，缺省值为[0]；LR 是学习速率，缺省值为 0.01。返回值 net 就是创建完毕的单层神经网络。

神经网络创建完毕后，可以用 train()函数进行训练。其调用格式为：

[net,tr] = TRAIN(NET,P,T,Pi,Ai)

其中 NET 就是上一步创建完毕的神经网络，P 是输入向量，T 是期望的非线性输出；Pi 是输入延迟条件，缺省值为 0；Ai 是网络的延迟条件，缺省值为 0。返回值 net 是训练完毕的网络，tr 是训练记录。

神经网络训练完毕后就可以用 sim()函数检验训练效果并预测新的非线性函数输出了。sim()函数也可以用于其他模型的仿真,前边在讲述控制系统仿真时曾经介绍过,这里再简要复习一下。其调用格式为:

[T,X,Y] = SIM('model',TIMESPAN,OPTIONS,UT)

其中'model'是模型描述,在这里就是训练好的神经网络 net(可以根据实际意义起另外的名称);TIMESPAN 是仿真时间间隔;OPTIONS 是仿真参数;UT 是外部输入,也就是用作检验或预测的非线性函数输入。返回值 T 是仿真的时间向量,X 是状态矩阵,Y 就是该仿真模型的输出。

8.2.2 非线性函数逼真

有了这些基本的神经网络仿真函数,我们就可以完成神经网络的一项重要功能——逼近非线性函数的工作了。请看下例。

[例 8.2] 测得某非线性环节的输入和输出向量如下。试构造一单层神经网络结构逼近此非线性环节,给出逼近的精度,并预测输入为 -1.2 时系统的输出。

输入 $P=(1.0\quad 1.5\quad 3.0\quad -1.2)^T$,输出 $T=(0.5\quad 1.1\quad 3.0\quad -1.0)^T$

分析:从输入和输出向量来看,是典型的非线性函数,在控制领域中一般要对其进行线性化。通常的线性化方法是最小二乘方法(参看本书第一章),需要编制较长的程序,并且精度不容易控制。下面我们根据 MATLAB 提供的函数,训练单层神经网络逼近这个非线性函数,并给出逼近精度。

结果:输入为 -1.2 时系统的输出为:

Y = -1.0466

最终的仿真精度为 0.0828。

求解过程:本例的求解分为以下几个步骤:

1. 绘制原输入和输出向量的误差平面

在 MATLAB Command Window 下键入下述语句:

```
%输入向量
P=[1,1.5,3,-1.2];
%输出向量
T=[0.5,1.1,3.0,-1.0];
%权值和阈值范围
w_range= -2:0.4:2;
b_range= -2:0.4:2;
%求解误差平面并绘图
ES=errsurf(P,T,w_range,b_range,'purelin');
plotes(w_range,b_range,ES);
```

相应的误差平面图形见图 8-10。

图 8-10 分为两部分,其中左边误差平面的三维图形,横纵坐标分别是阈值"Bias"和权值"Weight",高度坐标是误差平方之和。图中的三维曲面就是利用神经网络仿真时,选择相应的阈值和权值带来的误差平方值。从图中可以读出,这个误差平方值的范围是比较大的,阈值和权值在(-2,2)范围内时(见图中坐标或上文的代码),其值在(5,15)之间。

这说明该环节的非线性特性比较严重，用线性化的方法近似不可能完全消除逼近误差。我们的任务就是利用 MATLAB 提供的神经网络函数，在该误差平面中寻找误差最小的点，并将其作为该非线性环节的神经网络近似。

图 8-10 的右半部分是左边误差平面的辅助曲线，称作误差等高线。其横坐标是权值"Weight"，纵坐标是阈值"Bias"，图形中的白线表示使用该曲线上点的权值和阈值的神经网络拥有相同的误差。

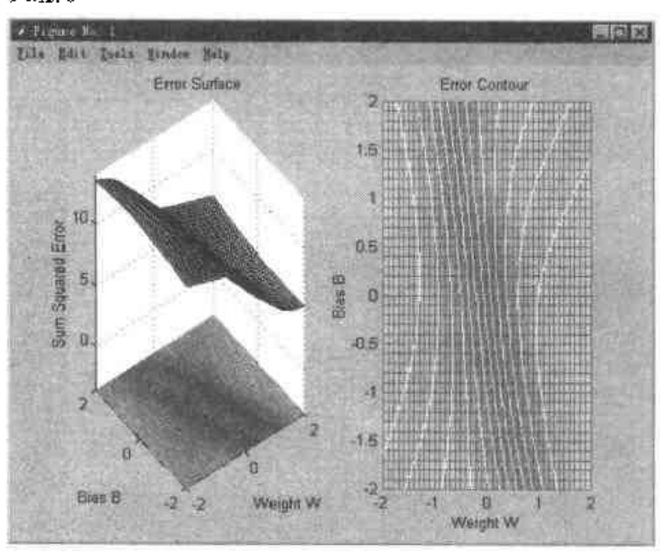


图 8-10 非线性函数误差图形

2. 创建并训练神经网络逼近此非线性环节

根据图 8-10 的误差平面和等高线，我们的任务就是寻求误差平面的最低点。MATLAB 提供的神经网络函数可以帮助完成此项功能。紧接上一问的 MATLAB Command Window，键入下述语句：

```
% 寻找最快学习速率
maxlr = maxlinlr(P,'bias');
%创建神经网络
net = newlin([-2,2],1,[0],maxlr);
%设定网络训练参数
net.trainParam.epochs = 7;
net.trainParam.goal = 0;
%训练神经网络
[net tr] = train(net,P,T);
%绘制训练轨迹
plotperf(tr,net.trainParam.goal);
grid;
```

上边的代码段中使用了一个新的函数 maxlinlr()，其功能是自动寻找 newlin()函数创建的神经网络的最快学习速率。其调用格式为：

lr = MAXLINLR(P,'bias')

其中 P 是神经网络的输入向量，'bias'表示包含阈值；lr 是返回的最快学习速率，可以

直接被 newlin()函数调用。

在设定网络训练参数时使用了两个神经网络参数.trainParam.epochs 和.trainParam.goal,分别表示训练的次数和训练目标。一般训练的次数是由实际的问题决定的,因为本例输入和输出数据都较少,所以选择较小的训练次数,否则有可能出现参数发散问题。训练目标可设为 0 或较小的正实数值。



图 8-11 是执行上边代码后得到的训练轨迹。

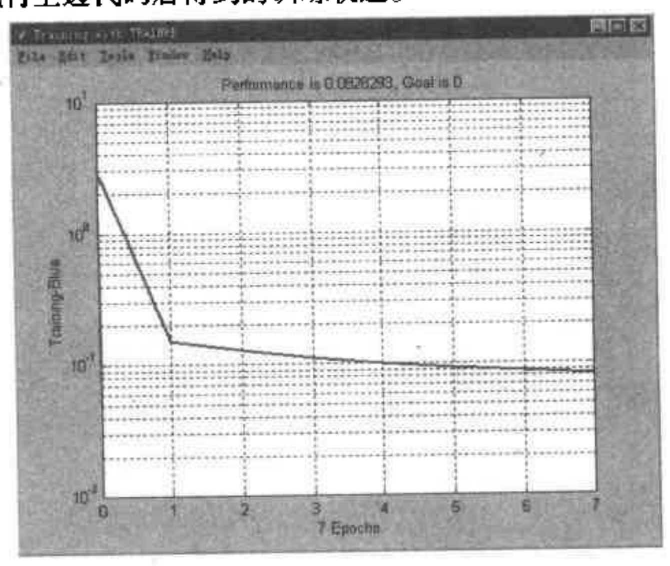


图 8-11 神经网络训练轨迹

图 8-11 的横坐标是训练的次数,纵坐标是训练的精度。从图中可以看出训练精度最终趋于 10^{-1}附近,从曲线的上方也可以读出"Performance is 0.0828293, Goal is 0",训练结果还是能够令人横意的。事实上,从该神经网络的训练轨迹来看,第一次训练改善最显著,精度从 3 跳变到 0.2 以下,以后的训练基本上都是在 10^{-1}附近微调。因此,综合时间和训练数果等因素来看,并不是说训练次数越多越好。此时在 MATLAB Command Window 下键入训练记录的变量名"tr",可以得到训练轨迹的数值:

```
tr =    epoch: [0   1   2   3   4   5   6   7]
        perf: [2.8650 0.1508 0.1284 0.1124 0.1010 0.0928 0.0870 0.0828]
```

3. 校验神经网络逼近结果

神经网络训练结束后,就可以使用该网络横拟此非线性环节了。题目要求求出输入为 -1.2 时系统的输出,可以使用 sim()函数求得:

```
P= -1.2
Y=sim(net,P);
```

结果为 Y = -1.0466,与期望值 -1.0 相差很小,验证了神经网络在逼近非线性函数方面是行之有效的。

此时还可以通过图形来验证神经网络的逼近效果,键入下述语句:

```
plot(P,T,'*',P,Y);
title('neural network approximation');
xlabel('input');
ylabel('output');
```

```
grid;
```

绘制的线性神经网络逼近曲线如图 8－12 所示。

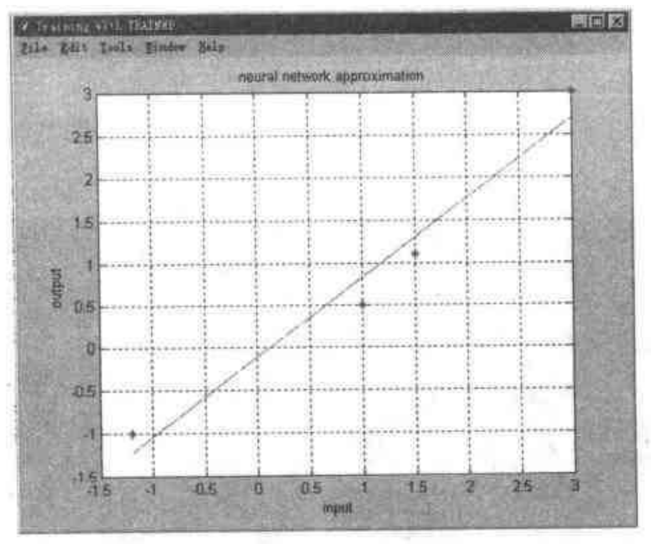


图 8－12　线形神经网络逼近曲线

图 8－12 中蓝色"*"号是原非线性环节的输入输出点，绿色实线是线性神经网络逼近的函数。从图中可以看出，该线性函数对此非线性环节的逼近是比较理想的。神经网络理论告诉我们，这是最小误差均方意义下的逼近，与最小二乘法得到的结果是基本类似的，仅仅存在非常小的误差。

小结：本例详细介绍了如何在 MATLAB 环境下使用线性神经网络逼近一个非线性环节。从目前的结果来看，除了程序编制比较简单之外还不能看出神经网络与传统的逼近技术相比有什么优点。这一方面是因为本例的数据量比较小，使用传统的逼近技术算法也不复杂；另一方面是因为我们使用的是线性神经网络的缘故。而对于那些数据量非常庞大、结构非常复杂的非线性系统来说，传统的逼近技术算法就有些力不从心，使用多层的非线性神经网络则可以得到很好的结果。

MATLAB 提供了包括非线性在内的各种神经网络模型，例如目前流行的前馈反向传播网络（Feed－Foreword Back Propagation network），实现函数为 newff（）；串级反向传播网络（Cascade－Forward Back Propagation network），实现函数为 newcf（）等等。利用这些函数可以逼近任意复杂程度的 L_2 空间非线性函数。下一节中我们将介绍一些与控制系统有关的常用神经网络模型，需要神经网络工具箱其他相关资料的读者可以查阅 MATLAB 帮助，这里就不再赘述了。

8.3　神经网络控制举例

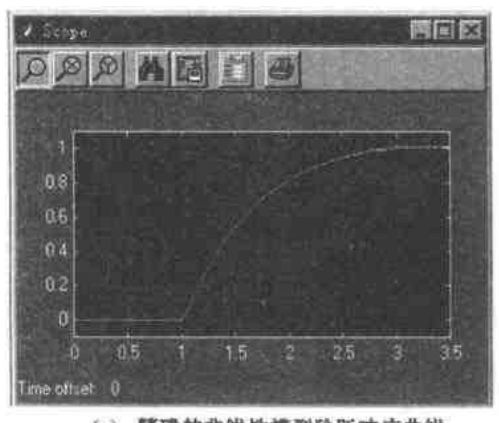
前边曾经概括地介绍过神经网络技术在控制领域的广泛应用，下面我们就以一个工程实际中的问题为例，具体介绍 MATLAB 环境下神经网络技术在控制领域的应用。

［例 8.3］　考虑某雷达天线仰角控制系统，通过改变直流电机电流可以控制天线臂角度 Φ。系统模型如下：

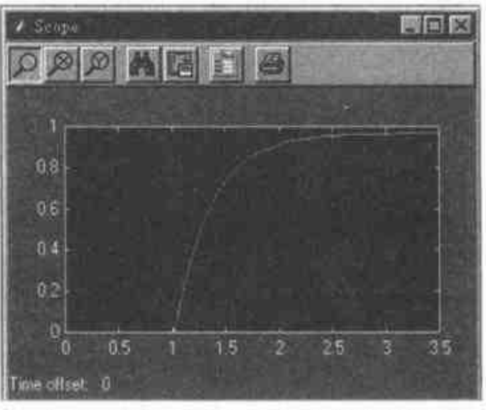
$$\begin{bmatrix} x_1' \\ x_2' \end{bmatrix} \begin{bmatrix} x_2 \\ 9.81\sin x_1 - 2x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u$$

此处 x1 是角位移,x2 是角速度。试利用合适的神经网络模型逼近此非线性系统,并比较相应的控制效果。

结果:按精确的非线性模型和近似前馈反向传播网络模型控制的阶跃响应曲线如图 8-13。



(a) 精确的非线性模型阶跃响应曲线



(b) 前馈反向传播网络模型阶跃响应曲线

图 8-13 系统的阶跃响应曲线

分析:对于本例这种结构比较复杂的完全非线性模型,再仿照上一节的单层线性神经元逼近的方法就很难得到满意的结果了。因此,我们采用较为复杂的前馈反向传播网络,通过多层网络来近似非线性函数。MATLAB 提供了一个用于前馈反向传播网络仿真的 newff()函数,其调用格式如下:

```
net = NEWFF(PR,[S1 S2...SNl],{TF1 TF2...TFNl},BTF,BLF,PF)
```

其中 PR 是输入向量的最小最大值矩阵;Si 是第 i 层网络的神经元数;TFi 是第 i 层网络的作用函数,缺省为 Sigmoid 型;BTF、BLF 和 PF 是网络相应的权重学习和训练函数,均可以使用 MATLAB 的缺省设置。

求解过程:

本例的求解分为以下几步:

1. 模型变换,分离出非线性函数

这样做一方面是为了控制方便,将线性环节和非线性环节分离,另一方面也是为了更好的利用神经网络逼近非线性函数的特性,使其不干扰线性环节。根据反馈线性化思想,取系统的参考模型为:

$$\begin{bmatrix} x_1' \\ x_2' \end{bmatrix} = \begin{bmatrix} x_2 \\ -9x_1 - 6x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ 9 \end{bmatrix} r$$

可得系统的非线性反馈环节:$f(X) = 9.81\sin x_1 - 2x_2$

相应的系统的控制规律为:$u = 9r - [9,6]X - f(X)$,其中 r 为设定值参考输入。

2. 利用前馈反向传播网络逼近系统的非线性环节

分离出系统的非线性环节以后,就可以利用神经网络对其进行逼近了。由系统的模型可知,所要设计训练的神经网络具有网个输入 x_1 和 x_2。网络设计目的是能够使其产生

逼近函数 f(x)的非线性输出,所以输出层只有一个输出神经元。中间选取一个两层神经元,Sigmoid 型采用作用函数,输出层取线性作用函数,在 MATLAB Command Window 下键入网络的源代码:

```
%系统状态变量初始化
x1 = [rands(1,300) * pi,rands(1,100) * pi];
x2 = [rands(1,300) * pi,zeros(1,100) * pi];
P = [x1;x2];
%离散化时间间隔
dt = 0.05;
%理想非线性输出
T = x2 + dt * [9.81 * sin(x1) - 2 * x2];
[R,Q] = size(P);
%第一层神经元数目
S1 = 10;
%第二层神经元数目
[S2,Q] = size(T);
pr = [0.7,1.05;0.95,1.04];
%初始化网络
net = newff(pr,[S1 S2])
xx1 = [(1:300)/100];
xx2 = [zeros(1,300)];
xx = [xx1;xx2];
y = xx2 + dt * [9.81 * sin(xx1) - 2 * xx2];
%循环次数
net.trainParam.epochs = 50;
%训练精度
net.trainParam.goal = 0;
%网络训练
[net,tr] = train(net,xx,y);
plotperf(tr,net.trainParam.goal);
grid;
```

在上边的代码中,我们采用包含两个隐层的前馈反向传播网络,第一层神经元数目为 10,第二层神经元数目为 1。同时取神经网络的训练次数为 50 次,要求的训练精度的目标为 0。此时相应的训练轨迹如图 8-14 所示:

从图中可以读出,训练 50 次以后达到的训练精度为 1.10172e-6,远低于上一节单层线性神经元的 10^{-1},这也是我们采用前馈反向传播网络的原因。可以预测,在这样的训练精度下,该神经网络对系统非线性环节的逼近一定是非常好的。下面我们通过一定的数据验证这个说法。在 MATLAB Command Window 下键入如下代码:

```
%数据初始化,取 60 个点
xx1 = [(1:30)/10];
xx2 = [zeros(1,30)];
xx = [xx1;xx2];
```

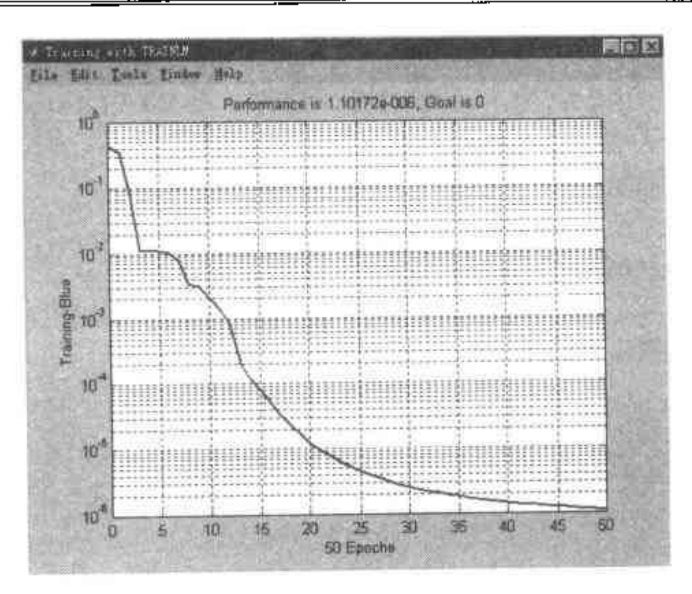


图 8-14　前馈反向传播网络训练轨迹

```
y = xx2 + dt * [9.81 * sin(xx1) - 2 * xx2];
%模型仿真
y2 = sim(net,xx);
%图形输出
plot(xx,y,'*',xx,y2);
title('neural network approximation');
xlabel('input');
ylabel('output');
grid;
```

此时系统的仿真曲线如图 8-15 所示。

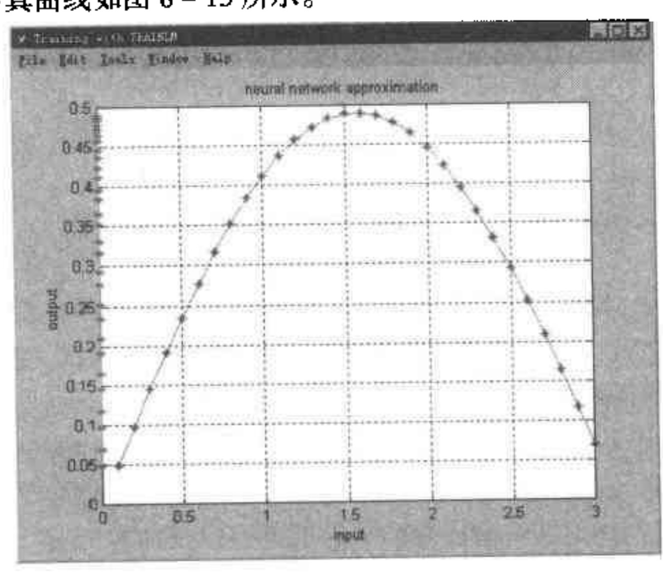


图 8-15　前馈反向传播网络逼近非线形函数轨迹

图中蓝色的"*"号是非线性函数的输出，红色的实线是神经网络的逼近曲线。可以看出，两条轨迹几乎完全重合，可以认为此时的神经网络已经完全实现了所要逼近的非线

性函数的功能。事实上，前边已经给出训练精度为 10^{-6} 数量级，在这样的精度下肉眼是很难判断两条曲线之间的差别的。

3. 控制效果仿真及对比

有了训练好的神经网络模型之后，就可以仿真评价其控制效果了。首先在 SIMULINK 下搭建原系统的方框图如图 8-16 所示。该方框图中包括三个“Sum”求和元件、两个“State-Space”系统状态空间描述元件、两个“Fcn”函数元件、一个“Step”阶跃输出元件、一个“Scope”示波器元件、一个“Matrix Gain”矩阵增益元件和一个“Mux”多路开关元件。对此方框图进行仿真，可以得到如图 8-13 系统的阶跃响应曲线(a)。然后就可以去掉图 8-16 中的两个“Fcn”函数元件，加入一个“MATLAB Fcn”函数元件，在该元件的设置对话框中填入代码：

```
y = sim(net,u)
```

对此方框图进行仿真，可以得到如图 8-13 系统的阶跃响应曲线(b)。

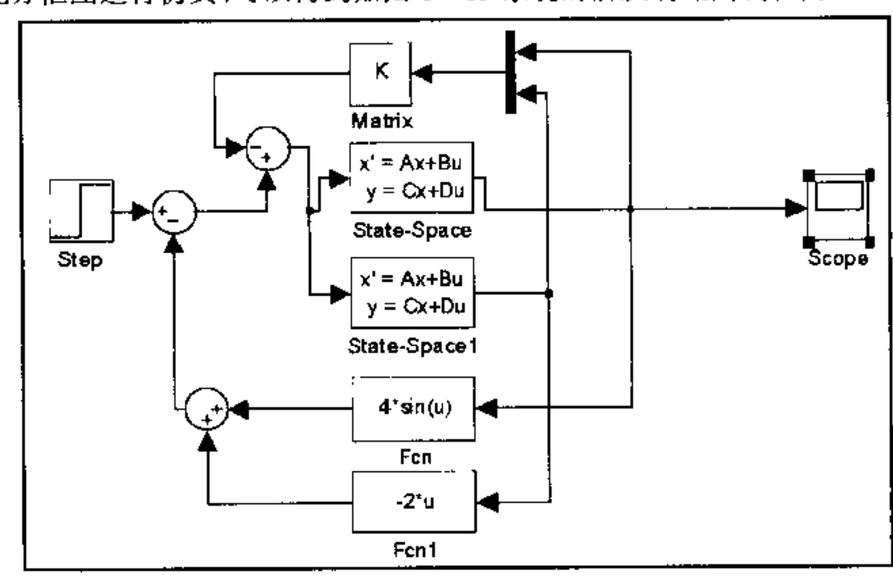


图 8-16 非线形系统仿真方框图

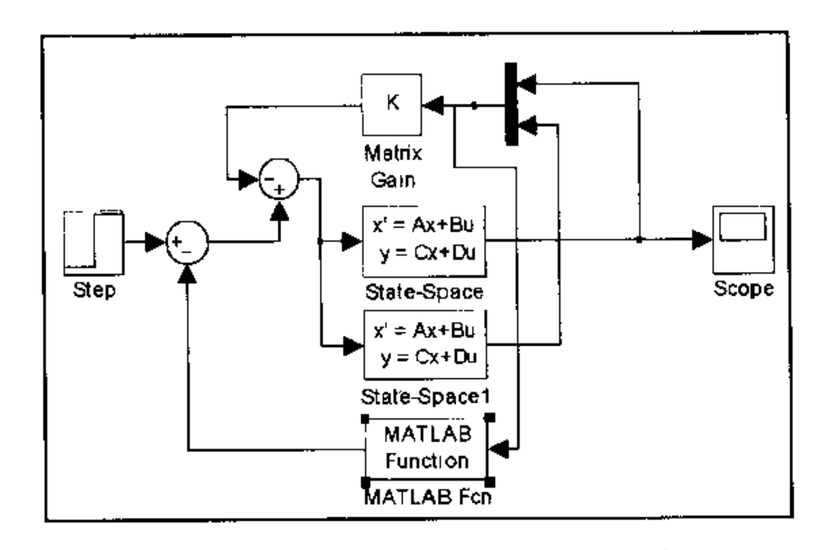


图 8-17 包含前馈反向传播网络的系统仿真方框图

对比这两幅阶跃响应曲线，可以看出曲线的形状和变化规律都基本一致，仅在细微的地方有所差别(注意两幅曲线的坐标刻度不完全一样)。这说明利用此前馈反向传播网络进行控制的效果是可以满意的。

小结：本例详细介绍了前馈反向传播网络在控制系统中的一个应用实例，给出了完整的 MATLAB 解法。实际运行了上边代码的读者会发现，利用包含前馈反向传播网络的方框图进行仿真比包含非线性环节的方框图仿真慢很多，这是因为每步仿真都要调用该神经网络，而神经网络的节点较多，对于计算机来说，显然不如简单的正弦函数计算简便。当然，相对于控制效果来说，这点开销是可以忍受的。

8.4 小　结

本章我们首先介绍了一些神经网络理论的基本概念及其在控制领域的应用，然后给出了 MATLAB 神经网络工具箱中一些常用的函数，最后在此基础上解决了一个实际的控制问题。简单地说，神经网络的核心就是利用简单的结点和复杂的拓扑结构逼近任意阶次的非线性函数。在控制领域，尤其是时延和非线性程度较高的流程工业控制问题中，利用神经网络进行控制也是当前的热点之一。

总之，神经网络在控制系统中的应用前景还是非常广阔的，MATLAB 强大的神经网络工具箱也一定会逐步更新、扩展，读者可以根据本书附录 C 中的 MATLAB 网址随时了解其最新动向。

附录 A　MATLAB 常用函数及控制工具箱命令

本附录搜集了 MATLAB 绝大多数常用的函数及语句命令，方便用户的查询。特别的，由于本书主要介绍的是 MATLAB 在控制系统中的应用，因此将控制系统工具箱中的命令也在此列出，本书前边各章节用到的命令和函数都可以在这里查询到。其他各工具箱的命令和函数请参阅附录 B。

表 A-1　通　用　命　令

• 管理命令和函数		
	help	在线帮助文件
	doc	装入超文本说明
	what	M、MAT、MEX 文件的目录列表
	type	列出 M 文件
	lookfor	通过 help 条目搜索关键字
	which	定位函数和文件
	demo	运行演示程序
	path	控制 MATLAB 的搜索路径
• 管理变量和工作空间		
	who	列出当前变量
	whos	列出当前变量(长表)
	load	从磁盘文件中恢复变量
	save	保存工作空间变量
	clear	从内存中清除变量和函数
	pack	整理工作空间内存
	size	矩阵的尺寸
	length	向量的长度
	disp	显示矩阵或文本
• 与文件和操作系统有关的命令		
	cd	改变当前工作目录
	dir	目录列表
	delete	删除文件
	getenv	获取环境变量值
	!	执行 DOS 操作系统命令

（续）

unix	执行 UNIX 操作系统命令并返回结果
diary	保存 MATLAB 任务
• 控制命令窗口	
cedit	设置命令行编辑
clc	清命令窗口
home	光标置左上角
format	设置输出格式
echo	底稿文件内使用的回显命令
more	在命令窗口中控制分页输出
• 启动和退出 MATLAB	
quit	退出 MATLAB
startup	引用 MATLAB 时所执行的 M 文件
matlabrc	主启动 M 文件
• 一般信息	
info	MATLAB 系统信息及 Mathworks 公司信息
subscribe	成为 MATLAB 的订购用户
hostid	MATLAB 主服务程序的识别代号
whatsnew	在说明书中未包含的新信息
ver	版本信息

表 A-2 操作符和特殊字符

• 操作符和特殊字符	
+	加
-	减
*	矩阵乘法
.*	数组乘法
^	矩阵幂
.^	数组幂
\	左除或反斜杠
/	右除或斜杠
./	数组除
kron	Kronecker 张量积
:	冒号
()	圆括号
[]	方括号
.	小数点
..	父目录

（续）

	...	继续
	,	逗号
	;	分号
	%	注释
	!	感叹号
	'	转置或引用
	=	赋值
	==	相等
	< >	关系操作符
	&	逻辑与
	|	逻辑或
	~	逻辑非
	xor	逻辑异或
• 逻辑函数		
	exist	检查变量或函数是否存在
	any	向量的任一元为真，则其值为真
	all	向量的所有元为真，则其值为真
	find	找出非零元素的索引号

表 A－3　基本数学函数

• 三角函数		
	sin	正弦
	sinh	双曲正弦
	asin	反正弦
	asinh	反双曲正弦
	cos	余弦
	cosh	双曲余弦
	acos	反余弦
	acosh	反双曲余弦
	tan	正切
	tanh	双曲正切
	atan	反正切
	atan2	四象限反正切
	atanh	反双曲正切
	sec	正割
	sech	双曲正割

（续）

asech	反双曲正割
csc	余割
csch	双曲余割
acsc	反余割
acsch	反双曲余割
cot	余切
coth	双曲余切
acot	反余切
acoth	反双曲余切
• 指数函数	
exp	指数
log	自然对数
log10	常用对数
sqrt	平方根
• 复数函数	
abs	绝对值
argle	相角
conj	复共轭
iamge	复数虚部
real	数实部复
• 数值函数	
fix	朝零方向取整
floor	朝负无穷大方向取整
ceil	朝正无穷大方向取整
round	朝最近的整数取整
rem	除后余数
sign	符号函数

表 A－4　基本矩阵和矩阵函数

• 基本矩阵	
zeros	零矩阵
ones	全"1"矩阵
eye	单位矩阵
rand	均匀分布的随机数矩阵
randn	正态分布的随机数矩阵
logspace	对数间隔的向量
meshgrid	三维图形的 X 和 Y 数组

（续）

:	规则间隔的向量
• **特殊变量和常数**	
ans	当前的答案
eps	相对浮点精度
realmax	最大浮点数
realmin	最小浮点数
pi	圆周率值 3.141 592 653 589 7 ……
i,j	虚数单位
inf	无穷大
nan	非数值
flops	浮点运算次数
nargin	函数输入变量数
nargout	函数输出变量数
computer	计算机类型
isieee	当计算机采用 IEEE 算术标准时，其值为真
why	简明的答案
version	MATLAB 版本号
• **时间和日期**	
clock	墙上挂钟
Date	日历
etime	计时函数
tic	秒表开始执行
toc	计时函数
cputime	CPU 时间（以秒为单位）
• **矩阵操作**	
diag	建立或提取对角阵
fliplr	矩阵作左右翻转
flipud	矩阵作上下翻转
reshape	改变矩阵大小
rot90	矩阵旋转 90°
tril	提取矩阵的下三角部分
triu	提取矩阵的上三角部分
:	矩阵的索引号，重新排列矩阵

表 A-5 特殊矩阵

	compan	友矩阵
	hadamard	Hadamard 矩阵
	hankel	Hankel 矩阵
	hilb	Hilbert 矩阵
	invhilb	逆 Hilbert 矩阵
	kron	Kronecker 张量积
	magic	魔方矩阵
	toeplitz	Toeplitz 矩阵
	vander	Vandermonde 矩阵

表 A-6 矩阵函数——数值线性代数

• 矩阵分析		
	cond	计算矩阵条件数
	norm	计算矩阵或向量范数
	rcond Linpack	逆条件值估计
	rank	计算矩阵秩
	det	计算矩阵行列式值
	trace	计算矩阵的迹
	null	零矩阵
	orth	正交化
• 线性方程		
	\ 和/	线性方程求解
	chol	Cholesky 分解
	lu	高斯消元法求系数阵
	inv	矩阵求逆
	qr	正交三角矩阵分解(简称 QR 分解)
	pinv	矩阵伪逆
• 特征值和奇异值		
	eig	求特征值和特征向量
	poly	求特征多项式
	hess	Hessberg 形式
	qz	广义特征值
	cdf2rdf	变复对角矩阵为实分块对角形式
	schur	Schur 分解
	balance	矩阵均衡处理以提高特征值精度
	svd	奇异值分解
• 矩阵函数		

（续）

	expm	矩阵指数
	expm1	实现 expm 的 M 文件
	expm2	通过泰勒级数求矩阵指数
	expm3	通过特征值和特征向量求矩阵指数
	logm	矩阵对数
	sqrtm	矩阵开平方根
	funm	一般矩阵的计算

表 A－7　泛函——非线性数值方法

	ode23	低阶法求解常微分方程
	ode23p	低阶法求解常微分方程并绘出结果图形
	ode45	高阶法求解常微分方程
	quad	低阶法计算数值积分
	quad8	高阶法计算数值积分
	fmin	单变量函数的极小变化
	fmins	多变量函数的极小化
	fzero	找出单变量函数的零点
	fplot	函数绘图

表 A－8　多项式函数

	roots	求多项式根
	poly	构造具有指定根的多项式
	polyvalm	带矩阵变量的多项式计算
	residue	部分分式展开（留数计算）
	polyfit	数据的多项式拟合
	polyder	微分多项式
	conv	多项式乘法
	deconv	多项式除法

表 A－9　通用图形函数

• 建立和控制图形窗口		
	figure	建立图形（图形窗口）
	gcf	获取当前图形的句柄
	clf	清除当前图形
	close	关闭图形
• 建立和控制坐标系		

（续）

	subplot	在标定位置上建立坐标系
	axes	在任意位置上建立坐标系
	gca	获取当前坐标系的句柄
	cla	清除当前坐标系
	axis	控制坐标系的刻度和形式
	caxis	控制伪彩色坐标刻度
	hold	保持当前图形
• 句柄图形对象		
	figure	建立图形窗口
	axes	建立坐标系
	line	建立曲线
	text	建立文本串
	patch	建立图形填充块
	surface	建立曲面
	image	建立图像
	uicontrol	建立用户界面控制
	uimenu	建立用户界面菜单
• 句柄图形操作		
	set	设置对象
	get	获取对象特征
	reset	重置对象特征
	delete	删除对象
	newplot	预测 Nextplot 性质的 M 文件
	gco	获取当前对象的句柄
	drawnow	填充未完成绘图事件
	findobj	寻找指定特征值的对象
• 打印和存储		
	print	打印图形或保存图形
	printopt	配置本地打印机缺省值
	orient	设置纸张取向
	capture	屏幕抓取当前图形

表 A-10 二维图形函数

• 基本 X-Y 图形		
	plot	线性图形
	loglog	对数坐标图形

（续）

	semilogx	半对数坐标图形（X 轴为对数坐标）
	semilogy	半对数坐标图形（Y 轴为对数坐标）
	fill	绘制二维多边形填充图
· 特殊 X－Y 图形		
	polar	极坐标图
	bar	条形图
	stem	离散序列图或杆图
	stairs	阶梯图
	errorbar	误差条图
	hist	直方图
	rose	角度直方图
	compass	区域图
	feather	箭头图
	fplot	绘图函数
	comet	星点图
· 图形注释		
	title	图形标题
	xlabel	X 轴标记
	ylabel	Y 轴标记
	text	文本注释
	gtext	用鼠标放置文本
	grid	网格线

表 A－11 语 言 结 构

· MATLAB 编程语言		
	function	增加新的函数
	eval	执行由 MATLAB 表达式构成的字串
	feval	执行由字串指定的函数
	global	定义全局变量
· 程序控制流		
	if	条件执行语句
	else	与 if 命令配合使用
	elseif	与 if 命令配合使用
	end	for，while 和 if 语句的结束
	for	重复执行指定次数（循环）
	while	重复执行不定次数（循环）
	break	终止循环的执行

（续）

	return	返回引用的函数
	error	显示信息并终止函数执行
• 交互输入		
	input	提示用户输入
	keyboard	像底稿文件一样使用键盘输入
	menu	产生由用户输入选择的菜单
	pause	等待用户响应
	uimenu	建立用户界面菜单
	uicontrol	建立用户界面控制

表 A-12　字符串函数

• 一般函数		
	strings	MATLAB 中有关字符串函数的说明
	abs	变字符串为数值
	setstr	变数值为字符串
	isstr	当变量为字符串时其值为真
	blanks	空串
	deblank	删除尾部的空串
	str2mat	从各个字符串中形成文本矩阵
	eval	执行由 MATLAB 表达式组成的串
• 字符串比较		
	strcmp	比较字符串
	findstr	在一字符串中查找另一个子串
	upper	变字符串为大写
	lower	变字符串为小写
	isletter	当变量为字母时，其值为真
	isspace	当变量为空白字符时，其值为真
• 字符串与数值之间变换		
	num2str	变数值为字符串
	int2str	变整数为字符串
	str2num	变字符串为数值
	sprintf	变数值为格式控制下的字符串
	sscanf	变字符串为格式控制下的数值
• 十进制数与十六进制数之间变换		
	hex2num	变十六进制数为 IEEE 标准下的浮点数
	hex2dec	变十六进制数为十进制数
	dec2hex	变十进制数为十六进制数

表 A－13 控制系统工具箱

• **建模**		
	append	追加系统动态特性
	augstate	变量状态作为输出
	blkbuild	从方框图中构造状态空间系统
	cloop	系统的闭环
	connect	方框图建模
	conv	两个多项式的卷积
	destim	从增益矩阵中形成离散状态估计器
	dreg	从增益矩阵中形成离散控制器和估计器
	drmodel	产生随机离散模型
	estim	从增益矩阵中形成连续状态估计器
	feedback	反馈系统连接
	ord2	产生二阶系统的 A、B、C、D
	pade	时延的 Pade 近似
	parallel	并行系统连接
	reg	从增益矩阵中形成连续控制器和估计器
	rmodel	产生随机连续模型
	series	串行系统连接
	ssdelete	从模型中删除输入、输出或状态
	ssselect	从大系统中选择子系统
• **模型变换**		
	c2d	变连续系统为离散系统
	c2dm	利用指定方法变连续为离散系统
	c2dt	带一延时变连续为离散系统
	d2c	变离散为连续系统
	d2cm	利用指定方法变离散为连续系统
	poly	变根值表示为多项式表示
	residue	部分分式展开
	ss2tf	变状态空间表示为传递函数表示
	ss2zp	变状态空间表示为零极点表示
	tf2ss	变传递函数表示为状态空间表示
	tf2zp	变传递函数表示为零极点表示
	zp2tf	变零极点表示为传递函数表示
	zp2ss	变零极点表示为状态空间表示
• **模型简化**		
	balreal	平衡实现
	dbalreal	离散平衡实现

(续)

	dmodred	离散模型降阶
	minreal	最小实现和零极点对消
	modred	模型降阶
• 模型实现		
	canon	正则形式
	ctrbf	可控阶梯形
	obsvf	可观阶梯形
	ss2ss	采用相似变换
• 模型特性		
	covar	相对于白噪声的连续协方差响应
	ctrb	可控性矩阵
	damp	阻尼系数和固有频率
	dcgain	连续稳态(直流)增益
	dcovar	相对于白噪声的离散协方差响应
	ddamp	离散阻尼系数和固有频率
	ddcgain	离散系统增益
	dgram	离散可控性和可观性
	dsort	按幅值排序离散特征值
	eig	特征值和特征向量
	esort	按实部排序连续特征值
	gram	可控性和可观性
	obsv	可观性矩阵
	printsys	按格式显示系统
	roots	多项式之根
	tzero	传递零点
	tzero2	利用随机扰动法传递零点
• 时域响应		
	dimpulse	离散时间单位冲激响应
	dinitial	离散时间零输入响应
	dlsim	任意输入下的离散时间仿真
	dstep	离散时间阶跃响应
	filter	单输入单输出 Z 变换仿真
	impulse	冲激响应
	initial	连续时间零输入响应
	lsim	任意输入下的连续时间仿真
	ltitr	低级时间响应函数

（续）

	step	阶跃响应
	stepfun	阶跃函数
• **频域响应**		
	bode	Bode（波德）图（频域响应）
	dbode	离散 Bode 图
	dnichols	离散 Nichols 图
	dnyquist	离散 Nyquist 图
	dsigma	离散奇异值频域图
	fbode	连续系统的快速 Bode 图
	freqs	拉普拉斯变换频率响应
	freqz	Z 变换频率响应
	ltifr	低级频率响应函数
	margin	增益和相位裕度
	nichols	Nichols 图
	ngrid	画 Nichols 图的栅格线
	nyquist	Nyquist 图
	sigma	奇异值频域图
• **根轨迹**		
	pzmap	零极点图
	rlocfind	交互式地确定根轨迹增益
	rlocus	画根轨迹
	sgrid	在 ω_n，z 网格上画连续根轨迹
	zgrid	在 ω_n，z 网格上画离散根轨迹
• **增益选择**		
	acker	单输入单输出极点配置
	dlqe	离散线性二次估计器设计
	dlqew	离散线性二次估计器设计
	dlqr	离散线性二次调节器设计
	dlqry	输出加权的离散调节器设计
	lqe	线性二次估计器设计
	lqed	基于连续代价函数的离散估计器设计
	lqe2	利用 Schur 法设计线性二次估计器
	lqew	一般线性二次估计器设计
	lqr	线性二次调节器设计
	lqrd	基于连续代价函数的离散调节器设计
	lqry	输出加权的调节器设计
	lqr2	利用 Schur 法设计线性二次调节器

(续)

	place	极点配置
• **方程求解**		
	are	代数 Riccati 方程求解
	dlyap	离散 Lyapunov 方程求解
	lyap	连续 Lyapunov 方程求解
	lyap2	利用对角化求解 Lyapunov 方程
• **演示示例**		
	ctrldemo	控制工具箱介绍
	boildemo	锅炉系统的 LQG 设计
	jetdemo	喷气式飞机偏航阻尼的典型设计
	diskdemo	硬盘控制器的数字控制
	kalmdemo	Kalman 滤波器设计和仿真
• **实用工具**		
	abcdchk	检测(A,B,C,D)组的一致性
	chop	取 n 个重要的位置
	dexresp	离散取样响应函数
	dfrqint	离散 Bode 图的自动定范围的算法
	dfrqint2	离散 Nyquist 图的自动定范围的算法
	dmulresp	离散多变量响应函数
	distsl	到直线间的距离
	dric	离散 Riccati 方程留数计算
	dsigma2	DSIGMA 实用工具函数
	dtimvec	离散时间响应的自动定范围算法
	exresp	取样响应函数
	freqint	Bode 图的自动定范围算法
	freqint2	Nyquist 图自动定范围算法
	freqresp	低级频率响应函数
	givens	旋转
	housh	构造 Householder 变换
	imargin	利用内插技术求增益和相位裕度
	lab2ser	变标号为字符串
	mulresp	多变量响应函数
	nargchk	检测 M 文件的变量数
	perpxy	寻找最近的正交点
	poly2str	变多项式为字符串
	printmat	带行列号打印矩阵

（续）

ric	Riccati 方程留数计算
schord	有序 Schur 分解
sigma2	SIGMA 使用函数
tfchk	检测传递函数的一致性
timvec	连续时间响应的自动定范围算法
tzreduce	在计算过零点时简化系统
vsort	匹配两根轨迹的向量

附录 B　MATLAB 常用工具箱函数

本附录搜集了 MATLAB 中与控制系统相关的常用工具箱函数及语句命令，方便用户的查询。事实上，用户也可以自己编写一些 MATLAB 函数，按照某种特定的要求生成自己的工具箱。

表 B－1 为工具箱目录索引，其余各表为工具箱函数的具体内容。有关控制系统工具箱的函数请参阅附录 A。

表 B－1　工具箱目录索引

工具箱名称	工具箱内容	表索引
Fuzz y	模糊系统	表 B－2
Ident	系统辨识	表 B－3
Image	图像处理	表 B－4
Local	局部函数库	表 B－5
Ncd	非线性控制设计	表 B－6
Nnet	神经网络	表 B－7
Optim	最优化	表 B－8
Robust	鲁棒控制	表 B－9
Signal	信号处理	表 B－10

表 B－2　模糊系统工具箱函数

• GUI 编辑器		
	Fuzzy	基本 FIS(模糊推理系统)编辑器
	mfedit	隶属度函数编辑器
	ruleedit	规则编辑器及(句法)分析程序
	releview	规则观察器及模糊推理框图
	surfview	输出曲面观测器
• 隶属度函数		
	dsigmf	两个“S”形隶属度函数的差
	gauss2mf	双边高斯曲线隶属度函数
	gaussmf	高斯曲线隶属度函数
	gbellmf	广义钟形隶属度函数
	pimf	π 形隶属度函数
	psigmf	两个“S”形隶属度函数的积

(续)

	smf	"S"形隶属度函数
	sigmf	"sigmoid(S)"形隶属度函数
	trapmf	梯形隶属度函数
	trimf	三角形隶属度函数
	zmf	"Z"形隶属度函数
• **命令行 FIS 函数**		
	addmf	将隶属度函数加到 FIS 中
	addrule	将规则加到 FIS 中
	addvar	将变量加到 FIS 中
	defuzz	去模糊隶属度函数
	evalfis	完成模糊推理计算
	evalmf	隶属度函数计算
	gensurf	产生 FIS 输出曲面
	getfis	获得模糊系统的特性
	mf2mf	在函数之间变换参数
	newfis	产生新的 FIS
	parsrule	分析模糊规则
	plotfis	显示 FIS 输入/输出图
	plotmf	显示出一个变量的所有隶属度函数
	readfis	从磁盘中装入 FIS
	rmmf	从 FIS 删除隶属度函数
	rmvar	从 FIS 中删除变量
	setfis	设置模糊系统特征
	showfis	显示带注释的 FIS
	showrule	显示 FIS 规则
	writefis	在磁盘中保存 FIS
• **高级技术**		
	anfis	Sugeno-type FIS 的训练程序
	fcm	利用 C 平均聚集方法找出簇
	genfis1	利用一般方法产生 FIS 矩阵
	genfis2	利用减法聚集方法产生 FIS 矩阵
	subclust	利用减法聚集方法估计簇中心

表 B-3 系统辨识工具箱函数

• **仿真和预测**		
	idsim	仿真一给定的系统
	pe	计算预测误差

(续)

	poly2th	从给定的多项式中构造矩阵
	predict	M 步超前预测
• 数据处理		
	dtrend	从数据集中删除方位
	idfilt	通过 Butterworth 滤波器对的数据进行滤波
• 非参数化估计		
	covf	估计数据矩阵的协方差矩阵
	cra	相关分析
	etfe	估计经验传递函数并计算周期图
	spa	频谱分析
• 参数估计		
	ar	利用各种方法 AR 信号模型
	armax	ARMAX 模型预测误差估计
	arx	ARX 模型的最小二乘估计
	bj	Box - Jenkins 模型的预测误差估计
	canstart	具有初值参数估计的多变量模型
	ivar	时间序列的 AR 部分的仪器 IV 估计
	ivx	单输出 ARX 模型的仪器可变估计
	iv4	ARX 模型近似最优的 IV 估计
	oe	输出误差模型的预测误差估计
	pem	一般线性模型的预测误差估计
• 建立模型结构		
	arx2th	ARX 模型的格式
	canform	正则形模型结构
	mf2th	将用户定义的模型结构封装入模型结构中
	modstruc	在 ms2th 函数中使用的模型结构
	ms2th	将标准状态空间参数封装入格式中
	poly2th	从给定多项式中产生矩阵
• 处理模型结构		
	fixpar	在状态空间 ARX 模型结构中，找出要修正的参数
	sett	在结构中设置取样间隔
	thinit	参数的(随机)初始值
	unfixpar	在状态空间和 ARX 模型结构中，放松参数
• 模型变换		
	th2arx	变格式模型为 ARX 模型
	th2ff	求模型的频率响应及标准偏差

（续）

	th2par	变格式为参数和协方差阵
	th2poly	求给定模型相应的多项式
	th2ss	变格式为状态空间表示
	th2tf	变格式为传递函数表示
	th2zp	求零极点、静态增益和标准偏差
	thc2thd	变连续时间模型为离散时间模型
	thd2thc	变离散时间模型为连续时间模型
• 模型表示		
	bodeplot	传递函数的 Bode 图或频谱
	ffplot	频域函数
	idplot	输入——输出数据
	nyqplot	传递函数的 Nyquist 图
	present	屏幕上的参数模型
	zpplot	零点和极点
• 信息提取		
	getmfth	获取定义模型结构的 M 文件的文件名
	getncap	获取数据点数和参数个数
	getff	选取频率函数
	gett	为某模型获取取样间隔
	getzp	th2zp 函数产生的零极点格式中，提取零点和极点
• 模型合法化		
	compare	将仿真和预测的输出与测量输出比较
	idsim	仿真一给定的系统
	pe	预测误差
	predict	M 步超前预测
	resid	计算和测试与某模型相关的留数
• 估计模型的不确定性		
	idsimsd	在仿真模型响应中说明不确定性
	th2ff	模型频率函数和标准偏差
	th2zp	零点、极点、静态增益及其标准偏差
• 模型结构选择		
	arxstruc	ARX 模型类的损失函数
	ivstruc	单输出类的输出误差拟合
	selstruc	根据各种准则选择模型结构
	struc	arxstruc 和 ivstruc 的典型结构矩阵
• 递归参数估计		
	rarx	对 AR 模型递归计算估值

（续）

	rarmax	对 ARMAX 模型递归计算估值
	rbj	对 Box - Jenkins 模型递归计算估值
	roe	对输出误差模型递归计算估值
	rpem	对一般模型递归计算估值
	rplr	对一般模型递归计算估值
	segment	分段数据并跟踪快变系统

表 B-4 图像处理工具箱函数

• 图像输入/输出		
	bmpread	从磁盘中读 BMP(Windows 下的位图)文件
	bmpwrite	将一 BMP 文件写入到磁盘
	gifread	从磁盘中读 GIF 文件
	gifwrite	将 GIF 文件写入磁盘
	hdfpeek	在 HDF 文件中列出目标标记/参考对
	hdfread	从 HDF 文件中读取数据
	Hdwrite	写数据到 HDF 文件中
	pcxread	从磁盘中读 PCX 文件
	pcxwrite	将 PCX 文件写入磁盘
	tiffread	从磁盘中读 TIFF 文件
	tiffwrite	将 TIFF 文件写入磁盘
	xwdread	从磁盘中读 XWD 文件
	xwdwrite	将 XWD 文件写入磁盘
• 实用程序		
	getimage	从坐标系中读取图像数据
	sbw	当图像为黑白图像时，其值为真
	isgray	当图像为灰度图像时，其值为真
	isind	当图像为加标图像时，其值为真
• 颜色操作		
	brighten	加亮或增暗一颜色板
	Cmunique	寻找唯一的颜色板及相应的图像
	Cmpernute	置换颜色板位置
	Cmgamma	校正颜色板
	Cmgamdef	缺省的校正表
	dither	Floyd – Steinberg 图像颤抖算法
	hsv2rgb	变 HSV 值为 RGB 颜色空间

(续)

	imadjust	调整并增强图像强度
	imapprox	利用更少颜色的图像逼近加标图像
	ntsc2rgb	变 NTSC 值为 RGB 颜色空间
	rgb2gray	变 RGB 值为灰度值
	rgb2hsv	变 RGB 值为 HSV 颜色空间
	rgb2ntsc	变 RGB 值为 NTSC 颜色空间
	Rgbplot	控制 RGB 颜色板分量的图形
• 几何操作		
	imcrop	修剪图像
	imresize	改变图像大小
	imrotate	旋转图像
	truesize	改变图像大小使之具有实际尺寸
	imzoom	放大或缩小图像和二维图形
• 图像增强/分析		
	brighten	增强或削弱颜色板
	grayslice	密度(强度)限幅
	histeq	直方图均衡化
	imadjust	调整和展宽图像强度
	imapprox	利用较少颜色的图像逼近图像
	imhist	图像直方图
	Impixel	一像素点的颜色
	Improfile	轮廓强度
	Imterp2	二维数据内插
• 图像统计		
	mean2	矩阵的均值
	corr2	二维相关系数
	std2	二维标准差
• 形态操作		
	bwarea	二进制图像中的目标区域
	dilate	加浓二进制图像
	erode	冲淡二进制图像
	edge	边界提取
	bweuler	欧拉数
	bwmorph	通过频率取样的二维
	bwperim	二进制图像中目标的周围
• FIR(有限冲激响应)滤波器设计		
	fsamp2	通过频率取样的二维 FIR 滤波器设计

（续）

	fspecial	特殊的二维滤波器设计
	ftrans2	通过频率变换的二维 FIR 滤波器设计
	fwind1	使用一维窗函数的 FIR 滤波器设计
	fwind2	使用二维窗函数的 FIR 滤波器设计
	imnoise	图像噪声
• **频率响应**		
	freqspace	二维频率响应的频率空间
	freqz2	二维频率响应
• **滤波**		
	colfilt	局部非线性滤波
	conv2	二维卷积
	filter2	二维滤波
	medfilt2	二维中值滤波
	mfilter2	屏蔽滤波
	nlfilter	局部非线性滤波
	wiener2	自适应二维维纳滤波
• **分块处理**		
	bestblk	分块处理的最佳块大小
	blkproc	按块处理 图像
	col2im	重新排列以形成图像
	colfilt	局部非线性滤波
	im2col	重新排列成列
• **个别区域**		
	mfilter2	屏蔽滤波
	roipoly	定义感兴趣的多边区域
	roicolor	用颜色定义感性趣的区域
• **变换**		
	dct2	二维离散余弦变换
	fft2	二维快速傅里叶变换
	fftshift	零频移到频谱中心
	idct2	二维逆离散余弦变换
	ifft2	二维逆快速傅里叶变换
	radon	Radon 变换
• **转换**		
	dither	Floyd - Steinberg 图像颤抖
	gray2ind	变灰度图像为附标图像

(续)

	hsv2rgb	变 HSV 值为 RGB 值
	im2bw	变图像为黑白图形
	imslice	在图像中获取/置入图像块
	ind2gray	变附标图像为灰度图像
	ind2rgb	变附标图像为 RGB 图像
	mat2gray	变矩阵为(灰度)图像
	ntsc2rgb	变 NTSC 值为 RGB 值
	rgb2gray	变 RGB 图像或值为灰度图像或值
	rgb2hsv	变 RGB 值为 HSV 值
	rgb2ind	变 RGB 图像为附标图像
	rgb2ntsc	变 RGB 值为 NTSC 值
• 图像显示		
	colorbar	显示颜色条
	colormap	设置或获取颜色查找表
	gray	线性灰度颜色板
	hsv，hot，jet	颜色板
	image	显示附标图像
	imagesc	数据定标并按图像显示
	imcontour	图像等高线
	immovie	制作图像动画
	imshow	显示所有类型的图像数据
	montage	按矩形剪辑方式显示图像
	subimage	显示多个图像
	warp	将图像卷成曲面
• 演示		
	imdemo	一般图像处理演示
	dctdemo	二维离散余弦变换图像压缩演示
	Firdemo	二维 FIR 滤波器演示
	nlfdemo	二维非线性滤波演示
• 专用函数		
	Cumsum3d	三维矩阵封装成二维矩阵时的累计和
	dct	一维离散余弦变换
	dctmtx2	一元二维离散余弦变换矩阵
	dithere	图像颤抖的 MEX 文件
	elem3d	三为矩阵封装成二维矩阵的元素位置
	getline	利用橡皮线跟踪鼠标移动
	getpts	利用可视点跟做鼠标移动

（续）

	getrect	利用橡皮矩形跟踪鼠标移动
	gif	压缩 GIF 数据
	hdfreadc	读 HDF 文件的 MEX 文件
	jdfpeekc	搜索 HDF 文件的 MEX 文件
	hdfwe	写 HDF 文件的 MEX 文件
	idct	一维逆离散余弦变换
	im2gray	变图像为灰度
	imhistc	图像直方图计算 MEX 文件
	ndx3d	三维矩阵封装成二维矩阵的索引
	rgb2im	变 RGB 图像为附标或强度图像
	rle	压缩数码数据
	tiff	压缩 tiff 编码数据
	vmquant	与彩色量化 MEX 文件接口的 M 文件
	waitbar	显示等待条
• MAT 文件		
	bwmorph.mat	bwmorph.m 文件的查找表
	forest.mat	Carmanah Old Growth Forest 的扫描相片
	mri.mat	人体心脑的磁性共振图像
	trees.mat	树的扫描图像

表 B-5　局部函数库

	matlabrc	MATLAB 的主启动 M 文件
	printopt	设置打印选项

表 B-6　非线性控制设计工具箱

• 对话框的管理		
	coneddlg	管理 NCD 工具箱固定编辑器的对话框
	paramdly	管理 NCD 油画参数的对话框
	rangedlg	管理坐标系范围的对话框
	refdlg	管理 NCD 参考信号的对话框
	stepdlg	管理 NCD 阶跃响应的对话框
	uncerdly	管理 NCD 不确定变量的对话框
• 主要界面		
	contrncd	建立 NCD 固定图形的用户界面控制
	menuncd	建立 NCD 固定图形的用户界面菜单
	ncdblock	包含 NCD 框图 SIMULINK 系统

（续）

	optblock	打开一个 NCD 图形的底稿文件
	optfig	建立一个 NCD 固定图形
• 主要优化技术		
	costfun	NCD 优化的代价函数
	nlinopt	执行优化算法
• 演示示例		
	ncddemo	包含所有 NCD 演示示例的 SIMULINK 系统
	ncddemo1	PID 控制器
	ncddemo2	带前馈控制器的 LQR
	ncddemo3	多输入多输出 PI 控制器
	ncddemo4	倒摆演示教程
	ncdtut1	控制设计示例
	ncdtut2	系统辨识示例
• 用户界面工具		
	dialog	主对话框建立 M 文件
	errordlg	建立出错对话框
	figflag	当图形为当前显示在屏幕上时其值为真
	helpdlg	显示一帮助对话框
	layout	定义对话框布局参数的底稿文件
	questdlg	建立提问对话框
	uiguide	有关用户界面约定/标准/建议的说明
	Warndlg	建立警告对话框
• 演示和教程实用工具		
	ncd1init	为 ncddemo1 的优化进行设置
	ncd2init	为 ncddemo2 的优化进行设置
	ncd3init	为 ncddemo3 的优化进行设置
	ncd4init	为 ncddemo4 的优化进行设置
	Penddate	为 ncdtut2（即倒摆）进行设置
• 界面实用工具		
	curobj	提供有关当前点的信息
	devidecb	将固定界分成两部分
	dcline	从 NCD 图中删除所有的图
	donep	收回 Close 按钮和菜单
	errorncd	管理 NCD 产生的常见错误
	fillaxes	建立约束边界并进行数据检测
	forceit	在已存在的界限内插入一子集

(续)

	keyncd	NCD 按键函数
	loadncd	装入并显示 NCD 数据
	makesurf	建立并限界曲面
	snapncd	以 22.5 间隔排出约束条
	refresho	使约束矩阵与图形一致
	saveload	当文件是从 SelectFile 中选择时，其值为真
	texted	收回 Port 可编辑的文本
	undoncd	放弃上次 NCD 图形用户界面的操作
	updatdlg	更新 NCD 对话框
• **最优化实用工具**		
	convertm	变约束矩阵为最优化格式
	miniipars	NCD 最小化分析
	montevar	初始化 Monte Carlo 仿真
	ncdglob	定义 NCD 全局变量
	str2mat2	变一行字符串为多行字符串
• **帮助文本文件(以.HLP 为扩展名)**		
	hotkey	热键帮助
	mainncd	一般 NCD 帮助
	paramdlg	最优化参数对话框的帮助
	Readncd	与 README.M 文件内容相同
	Stepdlg	阶跃响应对话框的帮助
	Uncerdlg	不确定性变量对话框的帮助

表 B-7 神经网络工具箱函数

• **误差分析函数**		
	errsurf	计算误差曲面
	plotep	在误差曲面上绘制权和基位置图
	plotes	绘制误差曲面图
• **δ 函数**		
	deltalin	对 PURELIN 神经元的函数
	deltalog	对 LOGSIG 神经元的函数
	deltatan	对 TANSIG 神经元的函数
• **设计**		
	solvehop	设计 Hopfield 网络
	solvelin	设计线性网络
	solverb	设计径向基网络
	solverbe	设计精确的径向基网络

（续）

• 初始化	
initc	竞争层初始化
initem	Elman 递归网络初始化
initff	至多三层的前向网络初始化
initlin	线性层初始化
initlvq	LVQ 网络初始化
initp	感知层初始化
initsm	自组织映射初始化
Midpoint	产生中点值
nwlog	对 LOGSIG 神经元产生 Nguyen - Widrow 随机数
nwtan	对 TANSIG 神经元产生 Nguyen - Widrow 随机数
randnc	产生归一化列随机数
randnr	产生归一化行随机数
rands	产生对称随机数
• 学习规则	
learnbp	反向演播学习规则
learnbpm	带预测的反向演播学习规则
learnh	Hebb 学习规则
learnhd	退化的 Hebb 学习规则
Learnis	内星学习规则
learnk	Kohonen 学习规则
learnlm	Levenberg - Marquardt 学习规则
learnlvq	学习矢量量化规则
learnos	外星学习规则
learnp	感知层学习规则
learnpn	归一化的感知层学习规则
learnwh	Widrow - Hoff 学习规则
• 矩阵	
combvec	创建所有的矢量集
delaysig	从信号矩阵中建立退化的信号矩阵
dist	计算矢量距离
ind2vec	变下标矢量为稀疏矩阵表示
normc	归一化矩阵列
normr	归一化矩阵行
pnormc	伪归一化矩阵列
quant	离散化成某数值的整数倍

(续)

	sumsqr	平方和
	vect2ind	变稀疏矩阵表示为下标矢量
• 邻域		
	nbdist	使用矢量距离的邻域阵
	nbgrid	使用栅格距离的邻域阵
	nbman	使用 Manhattan 距离的邻域阵
• 绘图		
	barerr	每个输出矢量的误差条形图表
	hintonw	绘制权值图
	hintonwb	绘制权值和偏差图
	ploterr	绘出网络误差与时间的关系
	plotes	绘制误差曲面
	plotfa	绘出目标模式及网络函数的逼近
	plotpv	绘出限幅神经元的感知器分类
	plotsm	绘制自组织映射图
	plottr	绘出网络误差记录及自适应学习速率
	plotvec	用不同颜色绘制矢量
• 仿真		
	simuc	竞争层仿真
	simuelm	Elman 递归网络仿真
	simuff	前向网络仿真
	Simuhop	Hopfield 网络仿真
	Simulin	线性层仿真
	Simup	感知层仿真
	Simurb	径向基网络仿真
	Simusm	自组织映射仿真
• 训练		
	Trainbp	利用反向演播训练前向网络
	px	利用快速反向演播训练网络
	trainc	训练竞争层网络
	Trainelm	训练 Elman 递归网络
	Trainlvq	训练 LVQ 网络
	Trainp	利用感知规则训练感知层
	Trainpn	利用归一化感知规则训练感知层
	Trainsm	利用 Kohonen 规则训练自组织映射
	trainwh	利用 Widrow Hoff 规则训练线性层
• 传递函数		

（续）

	compet	竞争层传递函数
	hardlim	硬限幅传递函数
	hardlims	对称硬限幅传递函数
	logsig	对数 S 型传递函数
	purelin	线性传递函数
	Radbas	径向基传递函数
	satlins	对称饱和线性传递函数
	tansig	正切 S 型传递函数

表 B-8　最优化工具箱函数

• 非线形最小化函数		
	attgoal	达到多目标
	constr	约束极小化
	fmin	无约束极小化(标量情况)
	fminu	利用梯度搜索的无约束极小化
	fmins	利用单纯形搜索的无约束极小化
	fsolve	非线性方程求解
	leastsq	非线性最小二乘
	minimax	极小极大求解
	Seminf	半定极小化
• 矩阵问题极小化		
	lp	线性规划
	qp	二次规划
	nnls	非负最小二乘
• 控制缺省值和选项		
	foptions	参数设置
• 演示		
	datdemo	数据拟合成曲线
	optdemo	演示菜单
	tutdemo	启动教程
	banademo	香蕉型函数的极小化
	goaldemo	目标达到
	dfildemo	有限精度滤波器设计
• 三次内插程序		
	cubic	内插 4 点以找出极大值
	cubici1	内插 1 点和梯度，以估计极小值

（续）

	cubici2	内插 2 点和梯度，以找出步长和极小值
	cubici3	内插 3 点和梯度，以找出步长和极小值
	cubici4	内插 4 点和梯度，以找出步长和极小值
• 二次内插程序		
	quad2	内插 3 点以超出极大值
	quadinter	内插 3 点以估计极小值
• 演示实用程序		
	eigfun	返回分类特征值的函数
	elimone	消去一变量
	filtfun	频率响应和根
	filtfun2	频率响应范数和根
	fitfun	返回拟合数据中的误差范数
	fitfun2	返回拟合数据中的误差矢量
• 半定实用程序		
	semifun	半定问题转换成约束问题
	findmax	在数据向量中内插极大值
	findmax2	在数据矩阵中内插极大值
	v2sort	分类两向量，然后删去丢失的元素
• 目标达到的实用程序		
	goalfun	目标达到问题转换成约束条件问题
	goalgra	变换目标达到问题中的梯度
• 测试程序		
	toptim	最优化测试组
	toptimf	最优化测试组的测试函数
	toptimg	最优化测试组的测试函数梯度
• 其它		
	graderr	用于检查梯度的不一致性
	lsint	初始化最小二乘程序的函数
	optint	初始化无约束极小化程序的函数
	searchq	线性搜索程序

表 B-9 鲁棒控制工具箱函数

• 可选系统数据结构		
	branch	从树中提取一分支
	graft	在树中增加一分支
	issystem	辨识一系统变量
	istree	辨识 -树型变量

(续)

	mksys	为系统建立树变量
	tree	建立树变量
	vrsys	返回标准系统变量名
• 建模		
	augss	系统增广(状态空间模型)
	augtf	系统增广(传递函数模型)
	interc	一般多变量内连系统
• 模型转换		
	bilin	多变量双线性变换
	des2ss	利用奇异值分解变系统为状态空间系统
	lftf	线性分式变换
	sectf	扇形变换
	stabproj	稳定和逆稳定映射
	slowfast	慢/快分解
	tfm2ss	变传递函数模型为状态空间模型
• 实用工具		
	aresolv	广义连续时间 Riccati 方程求解
	daresolv	广义离散时间 Riccati 方程求解
	riccond	连续时间 Riccati 方程的条件数
	driccond	离散时间 Riccati 方程的条件数
	blkrsch	通过 cschur 得到块有序实 Schur 形式
	Cschur	通过复旋转得有序复 Schur 形式
• 多变量 Bode 图		
	cgloci	连续特性增益轨迹
	Dgloci	离散特性增益轨迹
	Dsigma	离散奇异值 Bode 图
	Muopt	具有实/复数混合不确定性系统的 SSV(结构化奇异值)上界
	osborne	通过 Osborne 法求得的 SSV 上界
	perron	计算 Perron 特征值
	psv	Perron 特征结构的 SSV
	Sigma	连续奇异值 Bode 图
	ssv	结构化奇异值 Bode 图
• 因子分解技术		
	iofc	内外因子分解(列类型)
	iofr	内外因子分解(行类型)
	sfl	左边频谱分解

（续）

	sfr	右边频谱分解
• 模型简化方法		
	balmr	截断均衡模型简化
	bstschml	相对误差 Schur 模型简化
	bstschmr	相对误差 Schur 模型简化
	imp2ss	从脉冲响应到状态空间实现
	obalreal	有序均衡实现
	ohklmr	最优 Hankel 极小化逼近
	rechur	Schur 模型简化
• 鲁棒控制综合方法		
	h2lqg	连续时间综合
	dh2lqg	离散时间综合
	hinf	连续时间综合
	dhinf	离散时间综合
	Hinfopt	综合的迭代
	normh2	计算范数
	normhinf	计算范数
	lqg	LQG 最优控制综合
	ltru	LQG 闭环传递补偿
	ltry	LQG 闭环传递补偿
	youla	Youla 参数化
• 演示示例		
	accdemo	弹簧质量标准问题
	dintdemo	双积分器系统的 H_∞ 设计
	hinfdemo	飞机或大型空间结构的 H_2 或 H_∞ 设计示例
	ltrdemo	LQR/LTR 设计示例：飞机
	mudemo	μ 综合示例
	mudemo1	μ 综合示例
	mrdemo	鲁棒模型简化示例
	rctdemo	鲁棒控制工具箱演示——主菜单

表 B－10　信号处理工具箱函数

• 波形产生		
	sawtooth	产生锯齿波或三角波
	square	产生方波
	sinc	产生 sinc 或函数
	diric	产生 Dirichlet 或周期 sinc 函数

（续）

函数	功能
• **滤波器分析和实现**	
abs	取绝对值(幅值)
angle	取相角
conv	求卷积
Fftfilt	重叠相加法 FFT 滤波器实现
filter	直接滤波器实现
filtfilt	零相位数字滤波
filtic	filter 函数初始条件选择
freqs	模拟滤波器频率响应
freqspace	频率响应中的频率间隔
freqz	数字滤波器频率响应
grpdelay	平均滤波延迟(群延迟)
impz	数字滤波器的冲激响应
zplane	离散系统零极点图
• **线性系统变换**	
convmtx	卷积矩阵
poly2rc	从多项式系数中计算反射系数
residuez	Z 变换部分分式展开或留数计算
rc2poly	从反射系数中计算多项式系数
sos2ss	变系统二阶分割形式为状态空间形式
sos2tf	变系统二阶分割形式为传递函数形式
sos2zp	变系统二阶分割形式为零极点增益形式
ss2sos	变系统状态空间形式为二阶分割形式
ss2tf	变系统状态空间形式为传递函数形式
ss2zp	变系统状态空间形式为零极点增益形式
tf2ss	变系统传递函数形式为状态空间形式
tf2zp	变系统传递函数形式为零极点增益形式
zp2sos	变系统零极点增益形式为二阶分割形式
zp2ss	变系统零极点增益形式为状态空间形式
zp2tf	变系统零极点增益形式为传递函数形式
• **IIR 滤波器设计**	
Besself	Bessel(贝塞尔)模拟滤波器设计
Butter	Butterworth(比特沃思)滤波器设计
cheby1	Chebyshev(切比雪夫)I 型滤波器设计
cheby2	Chebyshev(切比雪夫)II 型滤波器设计
ellip	椭圆滤波器设计

（续）

	yulewalk	递归数字滤波器设计
• IIR滤波器阶的选择		
	buttord	Butterworth滤波器阶的选择
	cheb1ord	Chebyshev Ⅰ型滤波器阶的选择
	cheb2ord	Chebyshev Ⅱ型滤波器阶的选择
	ellipord	椭圆滤波器阶选择
• FIR滤波器设计		
	fir1	基于窗函数的FIR滤波器设计——标准响应
	fir2	基于窗函数的FIR滤波器设计——任意响应
	firls	最小二乘FIR滤波器设计
	intfilt	内插FIR滤波器设计
	remez	Parks－McCellan最优FIR滤波器设计
	remezord	Parks－McCellan最优FIR滤波器阶估计
• 变换		
	czt	线性调频Z变换
	dct	离散余弦变换(DCT)
	idct	逆离散余弦变换
	dftmtx	离散傅里叶变换矩阵
	fft	一维快速傅里叶变换
	ifft	一维逆快速傅里叶变换
	fftshift	重新排列FFT的输出
	hilbert	希尔伯特变换
• 统计信号处理		
	cov	协方差矩阵
	xcov	互协方差函数估计
	corrcoef	相关系数矩阵
	xcorr	互相关函数估计
	cohere	相关函数平方幅值估计
	csd	互谱密度(CSD)估计
	psd	信号功率谱密度(PSD)估计
	tfe	从输入输出中估计传递函数
• 窗函数		
	Boxcar	矩形窗
	triang	三角窗
	Bartlett	Bartlett(巴特利特)窗
	Hamming	Hamming(哈明)窗
	Hanning	Hanning(汉宁)窗

(续)

	blackman	Blackman(布莱克曼)窗
	chebwin	Chebyshev 窗
	kaiser	Kaiser 窗
• 参数化建模		
	invfreqs	模拟滤波器拟合频率响应
	Invfreqz	离散滤波器拟合频率响应
	prony	利用 Prony 方法的离散滤波器拟合时间响应
	stmcb	利用 Steiglitz – McBride 迭代方法求线性模型
	levinson	Levinson – Durbin 递归算法
	lpc	线性预测系数
• 特殊操作		
	rceps	实倒谱和最小相位重构
	cceps	倒谱分析和最小相位重构
	ate	降低序列的取样速率
	interp	提高取样速率(内插)
	resample	改变取样速率
	medfilt1	一维中值滤波
	deconv	反卷积和多项式除法
	modulate	通讯仿真中的调制
	demod	通讯仿真中的解调
	vco	电压控制震荡器
	specgram	频谱分析
• 模拟原型滤波器设计		
	buttap	Butterworth 模拟低通滤波器原型
	cheb1ap	Chebyshev I 型模拟低通滤波器原型
	cheb2ap	Chebyshev II 型模拟低通滤波器原型
	ellipap	椭圆模拟低通滤波器原型
	besselap	Bessel 模拟低通滤波器原型
• 频率变换		
	lp2bp	低通到带通模拟滤波器变换
	lp2hp	低通到高通模拟滤波器变换
	lp2bs	低通到带阻模拟滤波器变换
	lp2lp	低通到低通模拟滤波器变换
• 滤波器离散化		
	Bilinear	双线性变换
	Imprinvar	冲激响应不变法实现模拟到数字的滤波器变换

（续）

• 其它		
	conv2	二维卷积
	Cplxpair	将复数归成复共轭对
	Detrend	删除线性趋势
	fft2	二维快速傅里叶变换
	ifft2	二维逆快速傅里叶变换
	fiter2	二维数字滤波器
	pilystab	稳定多项式
	xcorr2	二维互相关

附录 C　MATLAB 相关网址介绍

在网络技术飞速发展的今天，通过 internet 获取相关的信息越来越受到人们的重视。无论身处何地，只要有一台计算机、一部调制解调器、一部电话，就可以在 internet 的汪洋大海中冲浪。下面我们就介绍一些与 MATLAB 相关的专业网站，在提供其 WWW 地址的同时做一些简要的介绍，以备读者查询。

1. Mathworks 公司网址

这是 MATLAB 的官方网站，MATLAB 就是由该公司推出的。其 http 超链接地址为：

http://www.mathworks.com

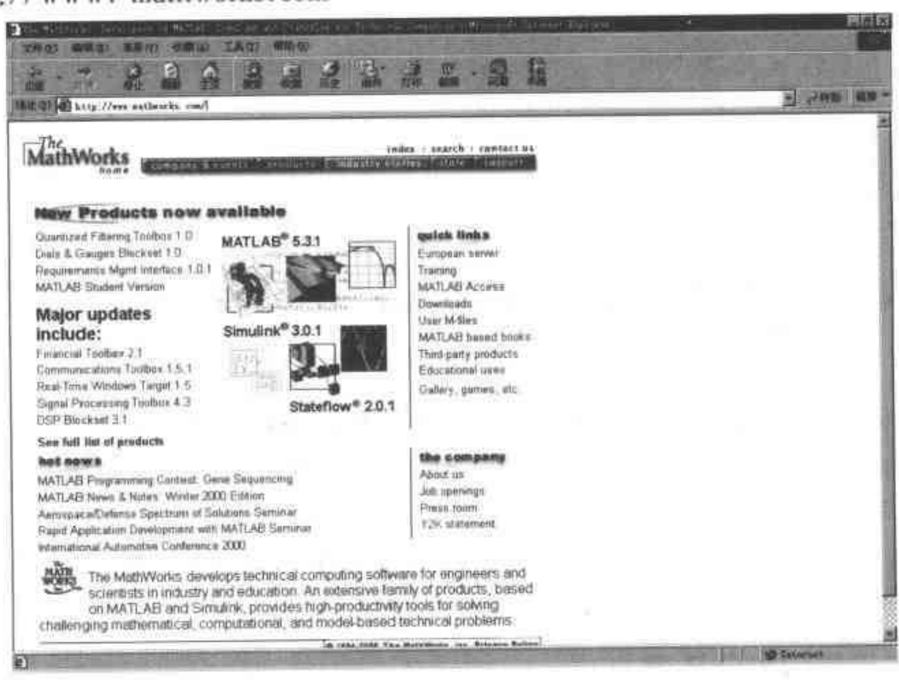


图 C－1　Mathworks 公司网址

从该网站可以获得有关 MATLAB 的各种最新的也是最权威的消息，包括新推出的工具箱、最新的 MATLAB 升级版本、各种技术支持、以及 Mathworks 公司的最新动态等等。当然，还有许多免费的 M 文件可供下载。

该网站包括以下几方面的内容：

1. Company & events——公司概况、最新动态以及发展方向；
2. Products——产品介绍，包括最新的升级版本和各种待开发的产品；
3. Industry stories——业界传奇，包括各种与 MATLLAB 相关的事迹等等；

4. Store——网上商店,在此可以购买、注册各种 MATLAB 产品;

5. Support——技术支持,有关 MATLAB 的各种疑问和要求可以在这里得到答复。

总之,对于关心 MATLAB 又有条件上网的读者,这是不可不去的地方。

2. Mathtools 公司网址

这是 Mathtools 公司的官方网站,其 http 超链接地址为:

http://www. mathtools.com

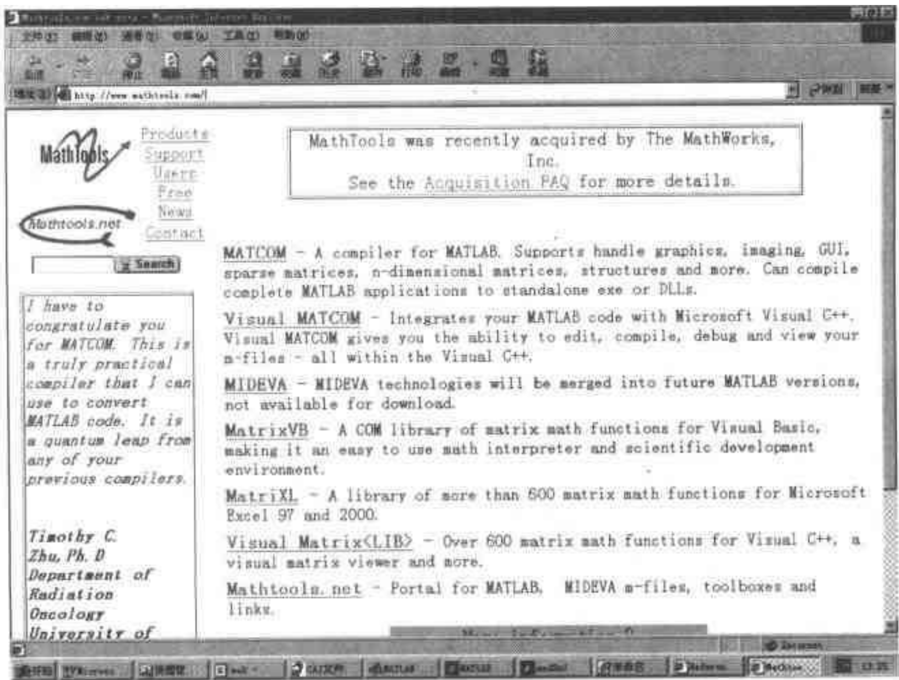


图 C-2 Mathtools 公司网址

Mathtools 公司的 MATCOM 软件是公认最优秀的 MATLAB 编译器之一,能够将包含图形句柄、图像、高维矩阵、结构体的 MATLAB 代码转换为标准的 exe 可执行文件或 DLL 动态链接库。

最近,Mathtools 公司已经被 Mathworks 公司并购了,但这个网址依然存在,对 MATLAB 编译器感兴趣的读者可以从这里获得需要的信息。

3. 应用数学软件乐园

这是一个使用 GB 码的 MATLAB 中文站点,其 http 超链接地址为:

http://202.102.229.60/grwy/matlab/index.htm

该网站是一个新开设的个人站点,因此涵盖的范围和深度都有待提高。不过其方便之处是服务器在国内一般情况下链接和下载都比较迅速;并且是为数不多的几家使用 GB 码的 MATLAB 中文站点之一,方便国内读者的阅读。

该网站比较有特色的栏目是"用 MATLAB 编写的三维地形生成算法",里面包含了许多个人编写的 M 函数,可以供浏览者免费下载。



图 C－3　应用数学软件乐园网址

